PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-283195

(43)Date of publication of application : 23.10.1998

(51)Int.Cl. G06F 9/46

G06F 9/46

G06F 9/06

(21)Application number : 09-164770

(22)Date of filing : 20.06.1997

(71)Applicant : MICROSOFT CORP

(72)Inventor : SHAW GEORGE H J

WOODRUFF BRYAN A

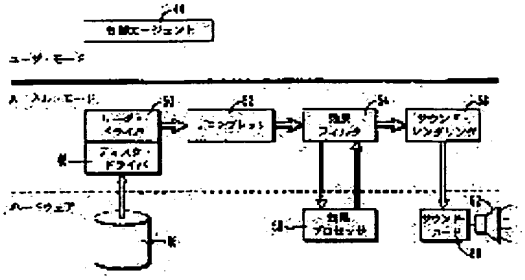
O'ROURKE THOMAS J

(30)Priority

Priority number : 97 825856 Priority date : 04.04.1997 Priority country : U S

(54) METHOD FOR MUTUALLY CONNECTING SOFTWARE DRIVER IN KERNEL MODE, COMPUTER PROGRAM PRODUCT, SYSTEM AND RECORD MEDIUM

(57)Abstract:



PROBLEM TO BE SOLVED: To improve system performance by mutually connecting software drivers by a standardized system and preventing the mode transition of an operating system in data processing time.

SOLUTION: The software drivers are mutually connected in the standardized system and the mode transition of the operating system is prevented in the data processing time.

In this system, a control agent 44 inquires the driver, performs mutual connection corresponding to a data format and a connection format and completely performs rendering in a kernel mode. Sound data are read from a disk drive 46 by a disk driver 48. A reader driver 50 is related in a vertical direction with the disk driver 48 and the reader driver 50 is mutually connected in a horizontal direction with a decompressor driver 52 as well and managed by the control agent 44.

LEGAL STATUS

[Date of request for examination] 14.05.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision

of rejection]

[Date of requesting appeal against examiner's

decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]

[Claim 1] In the approach for interconnecting a software driver and processing data efficiently by Carnell Mohd The step which opens one or two Carnell Mohd drivers or more, It is the step which forms one or two contact pin instances or more for connecting a driver. each contact pin instance is related with one of one or said the two drivers or more hierarchical -- having -- it is -- this -- with what is used between one or two drivers or more for data transmission The approach characterized by having the step which interconnects one or said two contact pin instances or more so that the continuation data flow pass which passes along one or said two drivers or more which operate by Carnell Mohd may be offered.

[Claim 2] It is the approach characterized by being created by expressing each contact pin instance with a file object, creating hierarchical relationship by specifying a related driver in an approach according to claim 1, referring to this driver as a file object of the I/O device which can be used on a system, and specifying this driver as the file object creation period of a contact pin instance as parents.

[Claim 3] The approach which refers to each of one or said two drivers or more, creates one or said two contact pin instances or more, and is characterized by having the step which interconnects further in an approach according to claim 1 after judging the type of the contact pin instance supported.

[Claim 4] At least one property set by which one or said two drivers or more were beforehand defined in the approach according to claim 1, It is shown in a third party component whether a method set and an event set are supported and which type of contact pin instance and a data format are supported by the driver. The approach characterized by making it possible for a third party component to form said contact pin instance, and to perform said interconnect between these contact pin instances.

[Claim 5] It is the approach characterized by to be included the bidirectional pin instance which transmits data to the driver connected from the related driver while receiving the data for processing by the input pin instance which receives the data for processing said contact pin instance by the related driver in an approach according to claim 1, the output pin instance which transmits data to the driver connected from the related driver, and the related driver.

[Claim 6] In an approach according to claim 1 said step which interconnects The step which receives the 1st reference to the 1st contact pin instance related with the 1st driver with a third party component for every pair of the pin instance which interconnected, The step which receives the 2nd reference to the 2nd contact pin instance related with the 2nd driver with a third party component, The step which hands over said 1st reference to said 2nd contact pin instance put on said 2nd driver with said third party component, It is the step which hands over said 2nd reference to said 1st contact pin instance put on said 1st driver with said third party component. This 1st and 2nd contact pin instance is an approach characterized by having what transmitted each other's data between each 1st [which was connected] and 2nd drivers.

[Claim 7] In an approach according to claim 1 said step which interconnects It is the step which receives the reference to an input pin instance with a third party component for every pair of the pin instance which interconnected. What received the data for processing this input pin instance by the receiving-side driver, It is the step which hands over said reference to the output pin instance put on the transmitting-side driver with said third party component. This output pin instance is an approach characterized by having the thing it was made to transmit to this input pin instance of said receiving-side driver to which data were connected from said transmitting-side driver.

[Claim 8] The computer read possible medium which has a computer executable instruction for performing the step indicated by claim 1.

[Claim 9] In the computer program product used by Carnell Mohd It is the computer usable medium which has the computer read possible program code means embodied so that other components of Carnell Mohd may be provided with the standardized interconnect mechanism. Said computer read possible program code means is equipped with the program code means for forming a contact pin instance. In order that said contact pin instance may transmit and receive data among other components of Carnell Mohd, while being used The computer program product characterized by interconnect with other contact pin instances put on other components being possible.

[Claim 10] The computer program product characterized by having further a program code means to answer enquiry from other components and to generate the information about the function of a contact pin in a computer program product according to claim 9.

[Claim 11] A program code means to generate contact pin instance information in a computer program product according to claim 10 is a computer program product characterized by having what mounted at least one of the property set accessed with another component, a method set,

and the event sets.

[Claim 12] The computer program product characterized by having further a program code means for controlling a specific contact pin instance by another component in a computer program product according to claim 9.

[Claim 13] A program code means to control a contact pin instance in a computer program product according to claim 12 is a computer program product characterized by having what mounted at least one of the property set accessed with another component, a method set, and the event sets.

[Claim 14] In the approach of making it possible to interconnect the 1st and 2nd device drivers, to standardize this device driver, and to communicate mutually by the extensible approach using the Carnell Mohd connection The step which provides the 1st device driver with a data format and a connection format with a third party component, The step which creates the handle to the connection which answered said third party component and acted to the 1st instance of said connection format as Ince Tan Scheidt by said 1st device driver, The step which returns said handle to said third party component by said 1st device driver, Said data format, said connection format, and the step that provides said 2nd device driver with said handle with said third party component, Answer said third party component and the 2nd instance of said connection format is formed by said 2nd driver using said handle. The approach characterized by having the step which enables said 1st driver to have and to transmit data to this 2nd driver within Carnell Mohd completely through said 1st and 2nd connection format instance.

[Claim 15] In an approach according to claim 14, it refers to said 1st and 2nd device drivers with a third party component. The step which judges which property set the device driver is supporting, The step which supplies the type of the connection which answers enquiry and each device driver supports to said third party component by said 1st and 2nd device drivers, The

approach characterized by having further the step which judges how connection between said 1st and 2nd device drivers is made with said third party component based on the supplied initial entry.

[Claim 16] In the approach of making it possible to interconnect the 1st and 2nd device drivers, to standardize this device driver, and to communicate mutually by the extensible approach using the Carnell Mohd connection It refers to said 1st and 2nd device drivers with a third party component. The step which judges which property set the device driver is supporting, The step which supplies the type of the connection which answers enquiry and each device driver supports to said third party component by these 1st and 2nd device drivers, The step which judges how connection between said 1st and 2nd device drivers is made with said third party component based on the supplied initial entry, The step which provides said 1st device driver with a data format and a connection format with said third party component, The step which creates the handle to the connection which answered said third party component and acted to the instance of said connection format to said 1st device driver as Ince Tan Scheidt by this 1st device driver, The approach characterized by having the step which returns said handle to said third party component by said 1st device driver, and the step which provides said 2nd device driver with said handle from said third party component.

[Claim 17] It is the Carnell Mohd data processing system which is two or more Carnell Mohd data-processing components containing the data source, and a generating component and a conclusion component, and is characterized by equipping said generating component with what reads the data sample of a data stream in the data source, and the thing for being the Carnell Mohd component interconnect between data-processing components, and carrying out routing of the data sample from a generating component to a conclusion component in the Carnell Mohd data processing system.

[Claim 18] In the Carnell Mohd Media rendering system The media source, They are two or more Carnell Mohd Media processing components containing a generating component and a conclusion component. A generating component reads the media sample of a media stream in the media source. A conclusion component carries out the rendering of said media stream. Each media processing component What has the contact pin instance for delivering a media sample between media processing components, It is the Carnell Mohd component interconnect between the media processing components created using the contact pin instance. The Carnell Mohd Media processing system characterized by having a thing for carrying out routing of the data sample from a generating component to a conclusion component.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] The field of this invention is in the thing belonging to development of a computer software driver. This invention relates to the tool which offers the standardized access point, a software framework, a regulation, etc. while simplifying the code generation of a driver. the software driver to which this invention was more specifically written by different developer -- Carnell Mohd -- interconnecting -- the user mode from Carnell Mohd -- or it is related with the standardized method which makes consecutive processing possible between different drivers, a computer program product, and DS, without performing inefficient transition to Carnell Mohd from user mode.

[0002]

[Description of the Prior Art] Usually it is made so that a software function may be offered, and a

software driver mitigates a system overhead, lessens constraint relatively, and is performed under the operating system so that hardware may be controlled. If it is made such, since the system code performed in order to guarantee normal actuation (behavior) and to carry out "the trap (trap)" of the interference with another process under activation under invalid access or an operating system will decrease, it has accelerated that a driver accesses hardware and serves a time critical processing demand (time critical processing request).

[0003] the level of operation or "Mohd (mode)" from whom the operating system has responded to the amount of access, and the security function mounted (mounting), and shoes differ -- **** -- it usually comes out that it is. For example, the usual application program was performed with the minimum priority, and has equipped the proper place fully with the security device for preventing interference with other applications. Only by hardware's letting the interface put under control pass, it is accessed. Although it is made for convenience the thing of explanation for which this is named "user Mohd (user mode)" generically, the Windows NT operating system currently used as a part of example of mounting of this invention which lower-** is supporting user Mohd. Similarly, the operating system of other most has a mode of operation equivalent to "user Mohd" regardless of the complexity.

[0004] On the other hand, the execution priority of a driver is far high, and since security protection is performed by Mohd of a low operating system, it is possible to access a driver to the actual hardware which a driver deals with directly in many cases. Much application is made into this thing [this being called "Carnell Mohd (kernel mode)" generally, and calling it such also on these specifications with the Windows NT vocabulary,], although it is more loose, and an advantage will be acquired if it is made to perform by performance-oriented Mohd more. Also in this case, other strong (robust) operating systems have equivalent Mohd functionally.

[0005] Since it has the intention of the general conception of a software driver controlling specific

hardware, a driver is separated from mutual, and is developed and, usually it is provided by the hardware manufacturer. The metaphor does not need to know the existence, if the software driver relevant to an add-in hardware card (add-in hardware card) which offers a certain kind of I/O service through the definition of a device does not need to communicate with all other drivers.

[0006] Since processing with desirable making it operate by Carnell Mohd operates [lack / to which the function which can communicate with such exclusive-like approach of development of a driver between drivers according to a situation relates / make / it / to make] by user Mohd, it happens to become disadvantageous in respect of performance.

[0007] As one of the typical examples of the present program which cannot use easily the Carnell Mohd driver used in this specification, a different processing block with which a user is called a filter is chosen, these are combined with one, and there is a graphing function to enable it to operate the stream of multimedia data continuously. data are a series of samples showing a sound or video -- typical -- a processing block -- elongation (decompression) of compressed data Processing and special effect (special effects) Processing and CODEC Rendering block (rendering block) which changes a function and data into an analog signal etc. -- it is.

[0008] The graphing functional division of a program interconnects, the operation is controlled, and the above filters are a user's input and relocation (rearrangement) of a processing block. It is typical to be placed by user Mohd so that it may respond. There is coordination in the stream of multimedia data, and since a lot of data are generated, performance has been an important problem. In general-purpose operating system, since system performance falls as a result of carrying out by repeating delivery and a change between user Mohd and Carnell Mohd and inspecting implicitly whether such transition (transition) is effective on security, a certain kind of multimedia application can be used.

[0009] Furthermore, since the processing block is connected to much time amount and hardware,

transition between components, which are user Mohd and Carnell Mohd, will be performed numbers of degrees. When such transition is performed, the overhead of another gestalt to which the total performance of a multimedia processing system is reduced will be included. If it changes between user Mohd and Carnell Mohd, the overhead related when copying data between different buffers arises, and this should not be produced if processing is left Carnell Mohd.

[0010] Another fault which happens when performing transition to user Mohd from Carnell Mohd is that the approach of carrying out scheduling of the activity task with an operating system is restricted. If it is possible to do an activity Carnell Mohd, since the twist performance-oriented scheduling approaches, such as a software interrupt and deferment procedure Kohl (deferred procedure call-DPC), can be used, system performance is optimized further.

[0011] Furthermore, it may be convenient if a different driver developer enables it to create a mutually connectable driver according to a common interconnect method and a common definition. If it does in this way, since every driver ability written to the common interface can interconnect in the system of a functional processing block, all data transition will be performed by Carnell Mohd. furthermore, the driver developer from whom many differ when well-known specification is used -- mutual -- it is operational and it becomes possible to develop the driver software which can interconnect.

[0012]

[Problem(s) to be Solved by the Invention] The purpose of this invention is improving system performance by interconnecting by the method which had the software driver standardized, and preventing the Mohd transition of an operating system at a data-processing period.

[0013] Furthermore, the purpose of this invention is offering the extensible base mechanism for software drivers which can be interconnected by the third party developer.

[0014] Another purpose of this invention is making it generate more performance critical processings in Carnell Mohd.

[0015] Furthermore, another purpose of this invention is that a third party component enables it to interconnect a software driver.

[0016] Although other purposes and advantages of this invention are indicated by the following explanation, the part has what is understood from the explanation, and the thing which can carry out learning by carrying out this invention. the instrumental the purpose and advantage of this invention are separately indicated to be by the claim -- the means and combination of RYUMENTO (instrument) realize and it is obtained.

[0017]

[Means for Solving the Problem] In order to attain the above-mentioned purpose, it is embodied below, and according to this invention currently explained to the wide sense, the approach and computer program product for interconnecting a software driver by Carnell Mohd are offered. Even if it does not depend for a certain given driver or filter on user Mohd's agent by supporting and defining the connection "a pin factory (pin factory)" which creates other pin instances (pin instance) put on other drivers, and the pin instance of a certain kind of type which can interconnect, consecutive processing of the processing message is made to be carried out by Carnell Mohd by the driver. If it does in this way, the flow of the data is completely carried out by Carnell Mohd, and since the overhead of crossing user Mohd becomes unnecessary, data will be processed efficiently.

[0018] The third party agent who wishes to connect a comp rye ANTO driver (compliant driver) refers for whether the function is in a driver. As such a function, he is Ince Tan Scheidt (instantiate) about a contact pin instance. It may say what kind of contact pin factory can be used for carrying out, and the related characteristic of the property of the input of the type of the data

processed, a data format, a transfer rate, a transfer medium or Mohd, and a contact pin instance or an output is included in this.

[0019] After a third party agent with typical performing by user Mohd makes a reference about the function of one or two comp rye ANTO drivers or more, the agent does "chaining" of two or more drivers to one, and determines the best connection characteristics by which data are processed the optimal between drivers. This decision step finishes enquiry of all driver ability, and after being able to choose the optimal traffic engineering standard, it is performed.

[0020] Next, a third party agent interconnects a driver by creating the instance of the required contact pin on each driver using a pin factory. An agent specifies a data format and a connection format as a part of contact pin instance creation. Furthermore, the reference to the contact pin instance created before is specified in the demand which creates other contact pin instances, and enables it to make connection between contact pin instances.

[0021] An actual contact pin instance is created with the gestalt of the instantiation-operation mounted under NT operating system by the creation I/O statement which returns the handle (handle) to a "file." This creation input/output request has included the driver instance handle to DS and reference which show a data format and connection format of a contact pin instance.

[0022] In order to create a comp rye ANTO driver, the driver developer is supporting a certain kind of standard function to which user Mohd's agent refers about a function, and enables it to perform interconnect between drivers. This is attained by use of "the set (set)" (that is, a property, a method, and the set of an event) which mounts the function to need with the gestalt of a certain operation currently built on the Windows NT operating system.

[0023] The set is logically defined as a thing with GUID (globally unique identifier) (globally unique identifier) which specifies the whole set, and RUID (relatively unique identifier) (it is a unique identifier, for example, an identifier relative within the set itself, relatively) of each

functional element in a set. Each set is related only with one or two IOCTLs(es) (IO Controls) (I/O control), and IOCTL is combined with the specification of a set and controls all interaction with a driver.

[0024] The set of three types is used with the gestalt of this operation. That is, they are a property set, a method set, and an event set. it is used in order that a property set may manage a value or the set points, such as sound volume and a transfer rate, within a driver, and Kohl is going to acquire the property value -- and/or, the flag which shows whether it is carrying out [which will set up a property value], and IOCTL single together are used. A method set is used in order to manage the operation which can perform a driver like allocation of memory, and Flushing (flushing) of a buffer, and it acts as Kohl of the specific method using single IOCTL. An event set is a device change notice and data starvation (data starvation). Like a notice, it is used in order to manage the event relevant to processing of a driver, and two IOCTL(s) are used. One is for making a specific event enable (enable), and another is a disable (disable) about a specific event. It is for carrying out.

[0025] In order to use a set, input/output control operation is started using the required data of the reference and others to DS with specific IOCTL, GUID of a set, and RUID. For example, when setting up a volume property by the sound card driver, IOCTL of a set property will be used, suitable GUID of the property set with the volume set point will be specified, it will direct that the pinpointing RUID in the set shows the volume property, and input/output control operation including the new volume set point will be performed.

[0026] In order to make a reference about the set supported, Nur GUID is used together with an enquiry flag by the specific set type specification IOCTL (for example, the property set IOCTL, the method set IOCTL, or event enabling (IOCTL)), and the list of sets GUID supported is returned. In order to make a reference about the property with which it was supported in a certain

set, a method, or an event, Set GUID, the set type IOCTL, and an enquiry flag are used together with the operation which returns the list of supported RUID.

[0027] Although the minimum function can be mounted in order to support a comp rye ANTO driver if a general-purpose set mechanism is used, expandability is still unrestricted. As long as the specific set is mounted, mutual actuation is possible for a set, and in order to create the system of the driver which can interconnect, it can be defined as the specification which can be coded independently by the driver developer from whom a large number differ. Furthermore, the property of the option which it not only can define an indispensable property with the need of supporting, a method, and an event, but can be mounted depending on the function and extension of a driver, a method, and an event can also be defined as this specification. It is also possible to incorporate an additional function by defining the set with an original driver developer other than the minimum fundamental similarity demanded, and assigning GUID to the set. If it enables it to enumerate the functions (that is, to hold enquiry of GUID supported and RUID) supported, a call side (caller), such as a third party control agent, can also perform suitable compensation depending on the function of a filter in which it also becomes a foundation to adjust expectation.

[0028] Although the above of this invention, and the other purposes and descriptions are as being indicated in detail by the following explanation and the claim, it is also possible to carry out learning by carrying out this invention explained below.

[0029] In order to show the approach how the above of this invention, and the other advantages and purposes are gained, below with reference to the gestalt of the concrete operation currently illustrated by the accompanying drawing, this invention explained briefly above will be explained in detail. An accompanying drawing is made to describe this invention more concretely and in detail, and to explain it while if the gestalt of typical operation of this invention was shown, it is not

alike and ****, therefore the range of this invention are not limited uses an accompanying drawing hereafter under comprehension.

[0030]

[Embodiment of the Invention] The vocabulary "user Mohd (user mode)" used on these specifications is the level of operation in an operating system in case the great portion of program written by the user is performed. User Mohd's level of operation has the highest security level, and in order to prevent interfering in a certain application program, application program with an another process, or a process, it is typical that a lot of overheads are consumed. Furthermore, access to a system resource is strictly controlled through a specific interface, and though an execution priority is not the minimum, it is general [the execution priority] that it is one of the minimum priorities.

[0031] The vocabulary "Carnell Mohd (kernel mode)" used on these specifications is the level of operation in an operating system with very less constraint than user Mohd's mode of operation. As Carnell Mohd's program, or an example of a process, it is the software driver (software driver) which controls a hardware component. It contains. As an example of representation, Carnell Mohd's program tended to be influenced by performance, therefore the overhead on actuation has become less than user Mohd's program. Furthermore, access to the system resource of hardware or many was not restrained, or its constraint has become less than the case of user Mohd's program far. The program code performed by Carnell Mohd trusts the order rightness of a programmer and conformity to a regulation (convention), and it is made for a system to operate good in many cases (for example, it is made not to destroy the address space of another program). as another vocabulary showing Carnell Mohd -- "-- a truss -- TEDDO (trusted) -- there is a" code.

[0032] The vocabulary "the driver (driver)" used on these specifications is a software driver

program with typical performing by Carnell Mohd. Vocabulary called a driver may mean the program part which divides and gives the actual executable program loaded on an operating system, or a function of a certain kind. In many cases, the driver is related with the hardware of a certain gestalt, but it comes out so and a certain need is not not necessarily.

[0033] Then, it is the thing of the functional division placed into the software driver, and in this, the driver itself is contained, and the vocabulary "the filter (filter)" used on these specifications is exhibited, in order that a connection point may transmit data through a filter. For example, the software driver may have one single function, if the filter with which some differ may be supported. Furthermore, it may connect with one internally and two or more filters from a different driver which opens the connection point to I/O externally may be collectively referred to as a single filter. moreover, vocabulary called a filter at more generic semantics -- elongation (decompression) etc. -- it may mean the operation performed like regardless of whether it happens with the software driver filter performed by Carnell Mohd, or it happens in another program code performed by user Mohd

[0034] The vocabulary "the driver object (driver object)" used on these specifications is the entity of the operating system defined by the operating system which manages a software driver and tells this as a system resource.

[0035] The vocabulary "the device object (device object)" used on these specifications is the entity of the system level defined by the system, and it defines whether this tells a part of function of the driver which can be used as a system resource, and can use the function and alien-system component of a driver. Both of driver objects and device objects are typically created at the time of loading of driver software, and initialization.

[0036] The vocabulary "the file object (file object)" used on these specifications is an entity of an operating system which manages the call of the resource specified by the device object, and this

is defined by the operating system. A file object offers the context about the operating condition of a driver object. Furthermore, a file object can be connected with another file object and a hierarchy target, if the former file object is specified as "parents" by the creation time of a new file object. A file object is used for management of all the I/O statements typically operated on a data stream.

[0037] The vocabulary "the data (data)" used on these specifications is all information processed through the filter of Carnell Mohd who interconnected. Although such data contain the media data showing video, an audio, a text, MIDI, etc., in the case of other applications, control information and a parameter may also be included. For example, although the Carnell mode filter graph may be used by process control operation, the control information delivered between different filters is used for the ******** sake there in the control signal which actuates machines. Although the example of a media processing system is shown, other applications can acquire an advantage from the system of the interconnect Carnell mode filter currently explained to this specification by the same approach.

[0038] Explanation of this invention is indicated by this specification within the context of the Windows NT (trademark) operating system currently offered from Microsoft (trademark). Furthermore, in order to understand the gestalt of the suitable operation currently explained to this specification, the knowledge of the I/O architecture of Windows NT is the requisite. input/output system -- and -- NT As a suitable document which explained the operating system at large, although there is Helen Custer work "the interior of Windows NT (Inside Windows NT)" (Microsoft Press issue), this book constitutes some of these specifications by citation.

[0039] Although it is explained by the following explanation of system entities, such as a driver and a file object, a device object, and a driver object, what kind of how to work these adopt on a Windows NT operating system, this invention can be mounted on other operating systems with a

similar component, and is unrelated to whether these operating systems are using the same vocabulary so that I may be understood, if this field becomes a familiarity person. For example, if another operating system has the entity which operates as a file object, the entity can be interpreted as a file object regardless of the actual name.

[0040] First, if it explains with reference to drawing 1 , the example of a system of illustration reads the stream of sound data in a disk drive, and is a rendering (rendering) about the sound data. It carries out and is made audible from a loudspeaker according to the conventional model. Although an certain amount of data is stored in the hard drive 20, this expresses the sound with the gestalt of the digitized sound sample. Other sources well-known in the digitization information sent as the source of a sound data stream using the telephone line, the digitization information from a network or other communication link packets, and this field otherwise can be considered. A data stream consists of digitization samples and time interval information is associated by the explicit time stump information by which these samples are added to a data format, a regulation, or each sample. Carnell Mohd's disk driver 22 interacts with the disk drive hardware 20, and is put under control of user Mohd's reader (reader) program component 24. a control agent (controlling agent) -- although a component which is different in order that 26 may perform the rendering of sound data is managed -- dynamic graphing function (dynamic graph building capability) When it has, a different software component is assigned dynamically and enables it to offer custom-made filtering or other processing paths according to assignment of an end user.

[0041] The reader component 24 interacts with a disk driver 22 using the standard-input/output control interface of an operating system, reads compression sound data in a disk drive 20, and is the address space (address space) of a user Mohd process. It serves to put into the buffer assigned by user Mohd as a part. Next, the decompressure (decompressor) component 28 elongates compressed data to the elongation (decompressed) format suitable for processing.

Like illustration, this whole step is performed by user Mohd using the process actuation (processbehavior) insurance mechanism of an attendant low priority.

[0042] the effectiveness filter (effects filter) 30 -- data -- actuation -- in addition, the effectiveness filter 32 of the attachment which a certain kind of special effect is acquired, and operates by Carnell Mohd -- **** -- it is. Furthermore, if the effectiveness processor 34 may exist, it may operate by the software by which the whole effectiveness filter emulates an actual hardware processor. In order to access the effectiveness filter 32, the effectiveness component 30 uses a system input/output control mechanism, and transmits data and control to an effectiveness filter. Also in this case, Carnell Mohd / user Mohd boundary is crossed, and this transition is performed.

[0043] The effectiveness filter 32 controls the effectiveness processor 34, and the need or the special effect made desirable is made on the data. This copies all data from the effectiveness component 30, and is performed by moving the copy to the effectiveness filter 32 again also at the effectiveness processor 34 depending on an actual system configuration. Although, as for many software effectiveness components, the hardware processor is associated, other components are functioning completely within the system software performed on a host processor.

[0044] If control and data are returned to user Mohd at the time of completion of processing of the effectiveness component 30, this will be transmitted to the sound rendering component 36 next. The sound rendering component 36 transmits control and data to the sound rendering driver 38, and in response, a sound rendering driver is processed and it controls a sound card 40 to carry out the rendering of the filtered data as a sound from a loudspeaker 42. As mentioned above, in order to recognize variety existence, the transfer between user Mohd and Carnell Mohd has made the rendering of sound data inefficient, so that I may be understood easily. Like a continuous sound or a video stream, if it lessens copying data numbers of degrees between

different buffers while multimedia data lessen transition of these inefficient nature and control for the property in which it is tended to influence timing, they are convenient.

[0045] The gestalt of 1 operation of this invention used on these specifications consists of services offered on Windows NT operating system architecture. This service is divided into the software component which the user of a system accesses and from which some differ. The 1st is user Mohd's API and this contains the routine for creating a contact pin instance and other file objects showing specific functions, such as a clock mechanism and a buffer allocation mechanism. Others are equipped fully with the routine and DS which support creating the driver to which a driver developer follows standardization architecture as a more important thing. If these functions currently prepared for the system are used, a different driver developer can create the comp rye ANTO driver which interacts mutually based on specific architecture. A user Mohd agent communicates with a comp rye ANTO driver through the environmental subsystem under activation by user Mohd who communicates with the system service of NT executive and an I/O manager. This is the same standard-input/output mechanism over other the I/O of all, and uses the existing system service as much as possible in this mounting of the gestalt of suitable operation.

[0046] The architecture of the system of drawing 1 using this invention is shown in drawing 2 . The control agent 44 refers to the driver told, interconnects according to a data format and a connection format, and is made to perform a rendering by Carnell Mohd completely. Furthermore, since a control agent receives the notice of an important event, he can control if needed. Termination of processing, the data starvation situation, etc. are included as an example of such an event.

[0047] With this configuration, sound data are read in a disk drive 46 by the disk driver 48, as mentioned above. The reader driver 50 controls a disk driver 48, and is related with the disk

driver 48 and the "perpendicular direction" according to NT layer (NT layered) I/O architecture using the former like the direction. The vocabulary of a "perpendicular direction" and a "horizontal direction" is used in order to distinguish the driver connection (perpendicular direction) made as a part of NT layer I/O architecture now, and the connection (horizontal) according to the interconnect Carnell Mohd driver dynamically performed by the third party control agent.

[0048] According to the connection method explained below, the decompressure driver 52 interconnects "horizontally" and the reader driver 50 is managed by the control agent 44. Decompressure 52 hands over data and control in the effectiveness filter 54, after developing by Carnell Mohd. An effectiveness filter hands over data and control to the sound rendering driver 58, after applying special effect using the effectiveness processor 56 if needed, and this driver controls a sound card and carries out the rendering of the data as a sound from a loudspeaker 62. Since the amount of overheads which usually happens will decrease if two or more transition between user Mohd and Carnell Mohd will be lost if processing is left Carnell Mohd and it processes by user Mohd so that I may be understood from drawing 2 , the advantage of an effectiveness side is acquired.

[0049] Next, it explains with reference to drawing 3 . Drawing 3 is the logic diagram showing the hierarchical property of the system object related to the interconnect software driver according to the gestalt of 1 operation of this invention. Since the software code image when being loaded to memory which can be performed is expressed, the driver object 64 is created. The driver code image includes the whole function of a driver, and the driver object 64 includes the information about images, such as a class of the location put on the system, and driver supported.

[0050] According to each type of the function which can be independently accessed by the control agent, it is device object 66a. -66N Created by I/O directory structure, these objects are available and express a different function accessed by user Mohd's client. As for these, it is

typical to express a filter or other functional divisions which can be used independently. The driver object 64 and device object 66a -66N As the enclosure box 68 shows, it is created at the time of install of a driver code.

[0051] Historically, a device object exists for every element of physical hardware. However, the flexibility of the newest input/output system enables a device object to express the filter completely mounted by software. Therefore, it is easy to create a device object for every instance of the filter chiefly mounted by software. Therefore, it is also possible to also mount so that, as for a software filter, each instance expressed with the device object may have the correspondence relation of 1 to 1 with a device object, and to mount so that the multiple-files object to which each expresses the client instance of a filter may be managed according to technique with a single more traditional device object.

[0052] At a device object and the example of illustration, it is device object 66a. Above, a file object is created and this expresses the independent instance of a function expressed with the device object. A device object expresses a filter and the file object expresses the actual instance of the filter used by the specific entity to managing two or more instances of the filter. Therefore, the file object 70 is device object 66a. It is the instance of the defined filter.

[0053] In order to use a filter, a control agent or other user Mohd clients open a file on a device available within I/O directory structure. The file object which has stored suitable context information is created, and the handle to the file is returned to a user Mohd client. He is the twin [as / whose all of a file object are the children of the same device object although a file object can be connected hierarchical by specifying a "parent" file object as creation time] (sibling). It also has relation.

[0054] The context information in a file object consists of "a condition (state) etc." of an entity etc. which the information which manages an input/output interface with a user Mohd client, and a file

object express. There is information which a system needs for context information, and custom [which can give still more nearly special semantics] possible area is also included. The example of the direction is shown in the explanation part of validation (validation) and the in PURIME tension of an IRP routing (routing) method which lower-** using custom possible area.

[0055] In order to offer a contact pin instance, the file object 70 showing a filter instance is used as parents, and the child file object showing the contact pin instance of a specific filter is created. To referring for the file object 70 about a definition and availability of a contact pin factory, an actual file object uses a specific file object as a suitable information context, it is created for every instance of a pin factory, and a contact pin instance is created effectively and correctly. For example, the file objects 72 and 74 express the contact pin instance of a filter expressed with the file object 70, and are connected with the file object 70 hierarchical. After it goes into a filter instance (expressed with the file object 70), the contact pin instance expressed with the file objects 72 and 74, respectively can be made the data pass which comes out from there, and it can use this in order to connect with other contact pin instances, when forming a series of chain filter or other driver ability.

[0056] If the file object of others [completely] similarly is connected with a pin instance and a hierarchy target with it being expressed with another file object to which a pin instance expresses a filter instance, and a file object with hierarchical relationship, and offering the context information on a pin instance and other functions are expressed, right context information will come to be acquired. Context information is required in order to distinguish a certain pin instance from other pin instances like a pin data format and a communication link type according to each parameter used for creation time.

[0057] Like a buffer allocation mechanism and a timing mechanism, an individual context or other actuation entities which require one of user Mohd's control through a handle can be

expressed with a file object. Furthermore, the hierarchical relationship between file objects (for example, buffer allocation mechanism related with the specific contact pin instance) is establishable by specifying a father-file object as the creation time of a child file object, if required. These parentages exist in order to determine the relation and structure between the file objects showing an actuation entity. Furthermore, since the "parent" file object of a particular type can make only a type of a certain kind of "child" file object, a creation validation mechanism is needed so that it may explain below. As for such a file object, also in this case, the corresponding handle has become available at user Mohd, and these handles are returned to a client through system API Kohl, such as NtCreateFile.

[0058] The handle to a file object is used by user Mohd clients, such as a control agent, in order to communicate with the Carnell Mohd driver. The hierarchical chain of a file object, a device object, and a driver object is an entry point (entry point) with which a I / O subsystem goes into return and an actual driver code to a driver object through a file object and a device object with hierarchical relationship. It enables it to reach. Such an entry point is the reference (for example, pointer) which points out the function in a software driver code. Furthermore, object path between a specific file object and a driver object with the entry point to a software driver code (pathway) Each of the object which is upwards also offers the reference to the DS used when carrying out routing of the IRP correctly according to routing and the validation mechanism which lower-** besides context information important when a I / O subsystem creates IRP.

[0059] The handle to a file object and an alien-system object is only for processes, and serves as a means when communicating with the object from which a user Mohd process serves as a foundation. What should be observed is being able to create two or more handles, in order to refer to the single system object used as foundations, such as a file object. It means that this can supply information to the pin instance to which two or more applications were expressed with the

file object.

[0060] As one of the information elements which become important when a different driver is interconnected, it is the depth (depth) of a device object stack. There is a parameter. This shows the IRP stack location of a specific driver object. If it does in this way, since an I/O manager is used and can be delivered between the drivers which interconnected using single IRP, IRP will be created separately, and performance improvement can be offered rather than it transmits it among various interconnect drivers. It is possible for each driver to let as an option Kohl, a suitable I/O manager, pass, to create new IRP for every continuation communication link, and to also make each new IRP transmit to the next driver on the chain of the driver which interconnected.

[0061] Next, it explains with reference to drawing 4 - drawing 6 . Drawing shows the extension of the system driver object which makes possible validation of file object creation of a different type, and routing of the input/output request packet (I/ORequest Packet:IRP) to a suitable handler, a device object, and a file object. Drawing 4 shows the driver object 76 showing the executable code which mounts one, or two a filter or other driver ability or more. Within a driver object, Windows NT architecture is demanding the reference to the creation handler which the software driver developer prepared. According to the gestalt of this operation, the multiplexing dispatch function (multiplexing dispatch function) 78 is referred to from the driver object 76 as a creation handler, and it is used in order to carry out routing to a specific creation handler according to the type of the file object which has a message created. The operation of the multiplexing dispatch function 78 is explained below with reference to the flow chart shown in drawing 8 .

[0062] It is possible to make these the same function according to how similarly other handlers of a driver object are mounted by showing a multiplexing dispatch function. In other words, referring to [of each type] the I/O handler (for example, reading writing, device control, etc.) used the

context information in the extension data in a device object, and a file object, and have pointed out the multiplexing dispatch function which carries out routing of the message to a suitable handler so that it may explain in detail below. By creation operation, the extension data in the device object which refers to a validation table are used, when there is no assignment of a father-file object. If there is assignment, the context information on a father-file object shows the right validation table.

[0063] By the driver developer, drawing 5 can be used by request and shows the driver object 80 with the specific device extension area 82 including the information only for drivers. It is called the file type validation table 84 to the location as which it defined in the device extension area 82 of the driver object 80, and the reference to DS including the string expression of the file object type 86 and the reference to the creation handler 88 by which each file type exception expressed is related are put on it. A creation multiplexing dispatch function uses the file type validation table 84, inspects the file object type created, and hands over control to a suitable creation handler after it. The explanation part of the following drawing 8 explains this in detail. The string inspected is ******** which is found out by IRP creation demand and used with user Mohd's NtCreateFile function Kohl from a file name string. NtCreateFile function Kohl is performed in a user Mohd function cel, in order to create a pin instance or other mechanisms.

[0064] Drawing 6 shows the file object 90 with the file context area 92 released in order that a software driver developer may use it. Reference is performed from the file context area 92 to the IRP demand handler table 94. The type with which the IRP demands 96 differ is related with the specific handler 98, and a suitable multiplexing dispatch function accesses a right handler using this information. In determining a right creation handler, the DS called the file type validation table 100 was referred to, and the reference 104 to the creation handler by which each file type exception expressed as the string expression of the file object type 102 is related is contained

there. In the case of the child file object (that is, file object which has not a device object but another file object as parents), the type is expressed with the string in the file object type 102, and the string compared. If a match is found, a related creation handler will be accessed using the reference related with the file object type string who was in agreement among reference 104. If a match is not found, since the demand is invalid, an error message will be taken out.

[0065] Next, if it explains with reference to drawing 7 , drawing shows the installation procedure for setting up creation validation and a mechanism. At step 106, an installation program creates the reference to a suitable multiplexing dispatch function in a driver object. As shown in drawing 4 , the creation handler has pointed out the general-purpose multiplexing dispatch function. Similarly, all referring to [other] the handler in the driver object 76 are other general purposes (generic) which have close relation with a specific handler. The multiplexing dispatch function is pointed out if needed. As an option, each referring to the handler can also point out the same multiplexing dispatch function, and in that case, this dispatch function can carry out routing of it to a suitable handler, after processing an IRP demand. Since the multiplexing function by this approach needs to receive demands (for example, creation, writing, etc.) of a different class, complicating is not avoided.

[0066] Next, each device object created as a part of install of a software driver executable code at step 108 is adjusted so that the file type validation table 84 may be referred to, as shown in drawing 5 . Processing of an IRP demand is started from a multiplexing dispatch function at step 110 by the last using the file type validation table 84 referred to from the suitable device object 80.

[0067] If a file object is created, the suitable IRP dispatch table 94 is created, and if required, it will be referred to together with the file object type validation table 100 by which the index was carried out. Creation of a file object type validation table is performed within the creation handler prepared according to the file object type. DS is created, this expresses the IRP dispatch table

94 and the file object type validation table 100, and the reference which points it out is stored in a specific location together with the file context information 92 on the specific file object 90 created.

[0068] Next, if it explains with reference to drawing 8 , drawing is a flow chart which shows the operation and its validation mechanism of a creation multiplexing dispatch function, and the interaction with the DS referred to from a system driver object, a device object, and a file object is also shown there. At step 112, a user Mohd process transmits the input/output request which creates a file object. This I/O creation demand is performed by calling the system API of NtCreateFile. At step 114, an I/O manager transmits IRP to the multiplexing dispatch function 78 based on the reference in the driver object 76 (refer to drawing 4).

[0069] If the multiplexing dispatch function 78 receives IRP of a creation demand, a test will be performed at step 116 and it will be judged whether there is any father-file object. Although information required in order to make this judgment is in the IRP itself, this is originally prepared by the user Mohd process. A user Mohd process prepares the handle which refers to a "parent" file object as a part of creation demand, and NT system creates IRP with refer to the right to a "parent" file object.

[0070] If there is no father-file object, the branch to the right is taken, and the multiplexing dispatch function 78 will use the device extension 82 from the suitable device object 80, and refer to the file type validation table 84 for it at step 118 (refer to drawing 5). The multiplexing dispatch function 78 inspects a file object type at step 120 by using the validation table 84 by comparing the string in a demand with the string of the file object type 86.

[0071] If it is judged that the string is in agreement at step 122, a suitable creation handler will be accessed at step 124. If not in agreement, a creation demand is refused at step 126. The creation handler accessed at step 124 creates a file object at step 126, or is made to create it. Using the created file object, a suitable creation handler creates "refer to [which points out the

IRP dispatch table 94 which was being created before] the file object" in the file context 92.

[0072] It is judged whether it returns to step 116 again and a father-file object exists. If a father-file object exists and it is [it was judged at step 116 and] contained in IRP relevant to a creation demand, the multiplexing dispatch function 78 will use the file context 92 from the father-file object 90, and refer to the IRP dispatch table 92 for it at step 130 (drawing 6). In a creation demand, refer to the file type validation table 100 for the multiplexing dispatch function 78 at step 132. Using the file type validation table 100, the multiplexing dispatch function 78 inspects a file object type at step 133 like the above by comparing the string in a demand with the string of the file object type 102.

[0073] If the string is in agreement and it will be judged at step 134, a suitable creation handler will be accessed at step 138. If not in agreement, a creation demand is refused at step 136. A suitable creation handler is used, a file object is created by 140, and a creation handler creates the reference which points out the IRP dispatch table 94 which created the new IRP dispatch table 94 of the file object created newly, and was created newly at step 142 in the file context area 92 of the file object 90 created newly. What should be careful of is that the same file object structure shown in drawing 6 in both cases is used, in order to explain an interaction with a father-file object and the child file object created effectively. Although the same structure exists in both cases (after a new file object was created), the information in which the usage differs and contains these also differs.

[0074] If a contact pin instance is created, contact pin ID is handed over and this shows the pin factory in the file "supports" creation of a pin instance always. Contact pin ID is possible also for inspecting as a string on a validation table completely the same with a file object being inspected, and that mounting approach is also various so that I may be understood, if this field becomes a familiarity person.

[0075] In order to make connection between different drivers, the common mechanism for confirming that a certain driver is supporting such interconnect is required. This common mechanism must make it possible to clarify a filtering function including a contact pin factory function. Furthermore, this kind of mechanism also needs a thing extensible so that a driver developer's flexibility may be improved.

[0076] A comp rye ANTO driver is defined, and in order to make it possible to clarify a function, one mechanism chosen with the gestalt of this operation is named the "set" of a related item. When this is used together with the existing I/O communication link mechanism, it is a convenient mechanism. The set is logically defined as a thing with GUID (globally unique identifier) which specifies the whole set, and RUID of each functional element in a set, relatively a unique identifier, for example, the relative identifier in the set itself. The DS of in order to carry out operation together with a set identifier and the selected RUID item, an and also [it is the need] is handed over as some input/output control Kohl as a parameter using a filter handle. In order to mount the perfect system of a function, it is enough just to assign a small number of IOCTL. Since the set of three different classes is established according to the function when mounted, needed IOCTL is a total of four pieces. How to use a set may differ in other mounting. Specific IOCTL notifies the selected element (RUID is used) to the handler of input/output control that it is interpreted or used by a certain approach. Furthermore, a control flag can be passed together with GUID and RUID, and control information can be specified in more detail.

[0077] The first set type is a property set and this is used together with the value or the set point placed in a driver and on related hardware. As an example of such the set point, there are a transfer rate, volume level (sound volume), etc. One IOCTL is related with a property set and a control flag is "get". A property command and "set" The property command is distinguished. If it does in this way, it can be used so that the same DS may set up or acquire a specific property,

and a driver can be determined based on IOCTL which had required action used. A right property is specified by the set identifier (set identifier) which consists of combination of unique GUID and RUID.

[0078] A method set is another set type used, and this is the set of action which can be performed by the driver. IOCTL required since a method set is specified is only one, and the right method to actuate is specified with the combination of unique GUID of a set identifier, and RUID. The method includes the function in which initialization for being used in order to control a driver, and using a driver, and a buffer are clear.

[0079] An event set is used in order to manage the event relevant to driver processing of a device change notice, the notice of data starvation, etc., and other notices defined by the convenient set when it was used with user Mohd application. Two IOCTL(s) are used, one is for enabling a specific event and another is for disabling a specific event. The DS required for the given event identified by RUID can be shared regardless of whether an event is enabled or it disables.

[0080] In order to use a set, input/output control operation is started using specific IOCTL and Set GID, and DS with RUID and the reference which points out other required data (for example, a control flag, DS, etc.). For example, setting up a volume property by the sound card driver is inevitably accompanied by using the pinpointing RUID in the set which shows suitable GUID of the control flag which shows the property set IOCTL and set property operation for input/output control operation, and the property set with the volume set point, and a volume property, and the new volume set point.

[0081] for making a reference according to a type about the set supported -- null -- IOCTL (for example, Property IOCTL, Method IOCTL, or event enabling (IOCTL)) with the control flag which shows listing of GUID and the set supported specific set type is taken out as a part of I/O

command, and the list of sets GUID supported is returned. In order to make a reference about the property, method, or event in a certain set supported, the control flag which shows Set GUID and listing of the element which set-type-IOCTL(s), and null-RUID(s) and is supported is used with an I/O statement. The list of RUID supported is returned as a result of an I/O statement. From this list, a third party agent can judge which option element (in a certain case) of the set mounted is supported.

[0082] the specification of the set uniquely identified by GUID -- both a driver developer and a third party control agent -- although -- the mechanism which can be used as a mounting guide is document-ized. A third party developer will get to know the function of a certain driver based on the response to enquiry, and will know the knowledge programmed in advance based on an abstract set definition. Similarly, a driver developer can use an abstract set definition as a guide when mounting the set or set group who offers the function got to know to what kind of third party agent.

[0083] In order to offer the connect function currently explained here, the comp rye ANTO driver must be supporting a set of a certain kind. The following tables are supported in a property set format, and show some important information which can be used when this invention is mounted. The 2nd table shows the property about the actual contact pin instance created as a template using a specific contact pin factory about the property about the contact pin factory where the first table is mounted with a filter.

[0084]

[Table 1]

（表１）

フィルタ・プロパティとその使い方	
プロパティ	説明
接続ピン・ファクトリ	特定のフィルタで作成できる異種タイプの接続ピン・インスタンスをリストしている。各区別可能なタイプはピン・ファクトリと呼ばれる。なお、これはこのデバイスでインスタンス生成できる接続ピン・インスタンスの総数ではなく、オーディオ入力やオーディオ出力のように、ユニークな接続ピン・タイプの数である。
接続インスタンス	ある接続ピン・ファクトリの、すでに作成されたインスタンスの数と、その特定接続ピン・ファクトリ用にサポートされるインスタンスの最大数をリストしている。フィルタが実際に接続されるまで総数が決定できないときは、このプロパティはａ－１を返す。
データ・フロー	接続ピン・ファクトリがフィルタに対して取り得るデータ・フローの方向をリストしている（例えば、フィルタに入る方向、フィルタから出る方向、またはフィルタから出入りする方向）。

[0085]

[Table 2]

（表１のつづき）

通信	ある接続ピン・ファクトリの通信要求をＩＲＰの処理の観点からリストしている。一部の接続ピン・ファクトリは相互接続できないが、グラフ上のソース・ポイントを表すデータのファイル・ソースへの「ブリッジ(bridge)」のように、関連づけられた他の形態の制御メカニズムを有している。ブリッジ制御メカニズムは情報がストアされているファイル名を間接的に設定することを可能にする。 実施の形態では、エージェント（接続ピン・インスタンスを作るときどのピン・ファクトリを使用するかを判断する）は接続ピン・ファクトリの「なし」、「シンク」または入力、「ソース」または出力、「両方」および「ブリッジ」通信タイプの本来の意味を理解できなければならない。例えば、ソース接続ピン・インスタンスはシンク接続ピン・インスタンスなどへのハンドルまたは参照を必要とする。 通信タイプのコンテキストでは、シンクとソースとはＩＲＰを処理するときの接続ピン・インスタンスの処置に関する。シンクは処理するＩＲＰを受信するのに対し、ソースはＩＲＰを次の適切な処理コンポーネント上に引き渡す。 シンクでもソースでもなく、接続グラフのエンド・ポイントを表している通信タイプが２つある。エンド・ポイントはデータが接続されたフィルタに出入りする場所を表している。「なし」とは接続タイプがインスタンシェイトできないことを意味するのに対し、ブリッジ通信タイプとは特定のプロパティを操作できるようにインスタンシェイトできるエンドポイントを意味する。例えば、ファイル・リーダの一部であるブリッジ・エンドポイントは処理されるデータをストアしているファイルのパスとファイル名を含んでいるプロパティをもっている可能性がある。

[0086] [Table 3]

（表１のつづき）

データ範囲	接続ピン・ファクトリがサポートできる、可能な限りのデータ範囲をリストしている。関連すれば、データのフォーマットも含まれる。一実施の形態では、データ範囲の配列があとに続くカウントは接続ピン・タイプがサポートできるが、プロパティの一部として使用される。この実装においては、異なるデータ範囲が異なるメディアまたはインタフェースの下でサポートされる場合（下記参照）、異なる接続ピン・ファクトリが特定フィルタで利用でき、この相違を受け入れることができる。さらに、各データ範囲構造はビット数やチャネル数といった、フォーマット固有の詳細用に拡張可能である。接続ピン・インスタンスが使用する実際のデータ・フォーマットはインスタンスの作成時に設定される。データ範囲プロパティは、その実際のデータ・フォーマットが特定の接続ピン・インスタンス用にどのようにされるべきかを判断するとき使用され、サード・パーティ制御エージェントによってアクセスまたは照会される。
インタフェース	特定の接続ピン・ファクトリ上のサポートされるインタフェースを示す他の集合ＧＵＩＤをリストしている。インタフェースは接続ピン・ファクトリを通して通信できるタイプまたはデータのタイプである。例えば、ＭＩＤＩデータ、ＣＤミュージック、ＭＰＥＧビデオなどは、フィルタが処理できる特定の規則とフォーマットをデータがもっているという意味でインタフェースとなる。このようなインタフェースはデータを発信するためのプロトコルも含んでいる。インタフェースはそれが通信されるときの媒体から独立している。
媒体	特定の接続ピン・ファクトリ上のサポートされる媒体をリストしている。媒体とは、ＩＲＰベース、ソケットなどのように、情報が転送されるときの方法またはメカニズムである。インタフェースは種々の異なる媒体の上に定義できる。本明細書で説明している好適実施の形態と実装例では、ＩＲＰベースの媒体とファイル入出力ベースの媒体が使用されている。

[0087]

[Table 4]

（表１のつづき）

データ交差	データ範囲のリストが与えられているとき接続ピン・ファクトリによって作られた最初の受付可能または「最良」データ・フォーマットを返す。このアプローチは、サード・パーティ・エージェントが異なるフィルタを１つにチェインニングするときデータ要件を判断できるようにするために使用される。一実装例では、データ交差(data intersection) プロパティはデータ範囲のリストの制約が与えられているとき接続ピン・ファクトリによって作られた最良データ・フォーマットを判断するために使用される。データ範囲のリストは前述したように接続される別のピン・ファクトリでデータ範囲プロパティを使用して取得できる。 サード・パーティ制御エージェントは、データ・タイプの詳細を知らないので、ある接続ピン・ファクトリのデータ範囲リストを使用し、現行接続ピン・ファクトリで「最良」（例えば、最初の受付可能データ・フォーマット）を返すことができる。２つの交差接続ピン・ファクトリの範囲のセットを返すことができるが、最良フォーマットだけがドライバによって返される。このようにすると、サード・パーティ制御エージェントはこの「最良」データ・フォーマットをグラフ内の次のドライバに適用して、仮想的な(virtual) 接続集合を作成してから、実際に接続ピン・インスタンスの作成を開始し、フィルタのグラフ全体を１つに接続することができる。これにより、制御エージェントはユーザによって選択された特定フィルタ・グラフの実行可能性を評価し、ユーザに起こる可能性のある問題を指摘してから、実際にグラフを接続することができる。返されるデータ・フォーマットはフィルタですでに行われている接続が与えられているとき、使用可能なフォーマットで制限することもできる。 このプロパティは実際の接続が行われるまで特定のデータ・フォーマットが決定できないときや、交差が異なる接続ポイント上の複数のデータ・フォーマットに依存するときエラーを返すことができる。基本的には、交差情報が提供され、プロパティ自身はデータ・フォーマットを返す。

[0088] [Table 5]

（表２）

接続ピン・インスタンス・プロパティとその使い方	
プロパティ	説明
状態	接続ピン・インスタンスの現状態を記述している。起こり得る状態には、停止、データ取得、データ処理、休止またはアイドルなどがある。状態は接続ピン・インスタンスの現モードを表し、ドライバの現在の機能とリソース使用状況を判断する。停止状態は接続ピン・インスタンスの初期状態であり、最小リソース使用状況のモードを表している。停止状態は、実行状態に到達するためにデータ処理のレイテンシ(latency) が最大であるポイントも表している。取得状態は、データがこの状態で転送されない場合でもリソースが取得されるモード（バッファ割当てなど）を表している。休止状態は最大リソース使用状況のモードと、実行状態に到達するまでの処理レイテンシがこれに応じて低いことを表している。データはこの状態で、転送または「プリロール(prerolled) 」できるが、これは実際には実行状態ではない。実行状態はデータが接続ピン・インスタンスで実際に消費または作成される（例えば、転送され、処理される）モードを表している。フィルタと基礎となるハードウェアの目的に応じてカスタム・プロパティを使用すると、コントロールの解像度を向上することができる。例えば、外部レーザ・ディスク・プレイヤをスピンアップさせるには、そのクラスに固有のある種のカスタム「モード」プロパティを設定することになる。このプロパティをセットすると、モードの効果に応じてデバイスの状態も変更されるが、必ずしもそうとは限らない。

[0089]

[Table 6]

(表２のつづき)

優先度	フィルタによって管理されるリソースへのアクセスを受け取るための接続ピン・インスタンスの優先度を記述しており、リソース割当ての調停(arbitration)においてフィルタによって使用される。このプロパティがサポートされていれば、サード・パーティ制御エージェントは限定リソースをこの特定接続およびインスタンスと共用できる他のすべてのドライバの他のすべての接続ピン・インスタンスに相対的な特定ピン・インスタンスの優先位置を指定することができる。この優先度プロパティが実装されていると、エージェントは単一優先度クラス内の優先度をもっときめ細かくチューニングすることができる。例えば、優先度はある種のサブクラスを関連づけることができる。同一リソースを競合する２つのドライバが同一優先度クラスをもっていれば、サブクラス優先度は２ドライバ間のリソース割当てを決定するために使用される。サブクラス優先度も同一であれば、調停により、最初の接続ピン・インスタンスがリソースを受け取ることになる。
データ・フォーマット	接続ピン・インスタンスのデータ・フォーマットを取得または設定するために使用される。

[0090] In order to create connection between different drivers so that I may be understood, if this field that is not what the above-mentioned table is mere instantiation and is limited to this becomes a familiarity person, it is possible to mount the property with which a large number differ, and a schema. the capacity to mount the property set with same driver manufacturer or development group who one important element is a standardization multiplier (standardization factor), and is different -- **** -- since it is, it is enabling it to create the driver which can interconnect.

[0091] Another useful property set gives the topology information about the internal relation of the contact pin factory of the input on a certain filter, and an output. Not only the relation between the input pin factory on a certain filter, and a corresponding output pin factory but this information has indicated processing [what kind of type] is performed between the pin factories of an input and an output. As an example of the processing performed, there are different data conversion, data elongation, an echo denial (cancellation), etc. If you use such information by the automation filter graphing function which creates an actual contact pin instance and connection after it traces

the connection pass on an assumption using two or more filters, it is convenient. Fundamentally, topology information explains the internal structure of a filter and opens this to enquiry from a third party agent through a property set mechanism.

[0092] Therefore, the comp rye ANTO driver mounts the specified property set. Therefore, a third party control agent can perform enquiry and a setup to a comp rye ANTO filter, when it turns out that a certain property set is supported. A whole target is acquiring sufficient information about the approach of connecting a different filter to one and making a filter graph.

[0093] Although the minimum function is mounted and the system of a comp rye ANTO driver can be supported if a general-purpose set mechanism is used, expandability is unrestricted even in such a case. as long as, as for the set, the specific set is mounted -- mutual -- in order to create the system of the driver which can interconnect [that it is operational and], a definition can be given in the specification which can be independently coded by the driver developer from whom a large number differ. Furthermore, the property of the option which it not only can define the indispensable property which must be supported, a method, and an event, but can be mounted according to driver ability and extension, a method, and an event can also be defined as this specification. It is also possible to incorporate an additional function by defining the set with an original driver developer other than the fundamental similarity required of the minimum, and assigning these GUID.

[0094] Next, it explains with reference to drawing 9 and drawing 10 . Drawing graphic-form-izes the process which connects two Carnell mode filters, and shows it. Drawing 9 is a logic-block explanatory view, and each filter instance and a contact pin instance are expressed with the file object there. Drawing 10 is a flow chart which shows the step when creating a file object and suitable connection.

[0095] It starts from step 144 and the instance of a filter A146 and the instance of a filter B148

are created by the user Mohd agent. These are created with a specific device using the standard file system API which creates a file. It is because it refers to the function of each filter about the contact pin factory which that the filter A146 and the filter B148 serve as a comp rye ANTO filter or a driver mounted the property with these suitable, the method, and the event set, supported creation of a contact pin instance, and was defined the set supported and for [its] filters.

[0096] A third party control agent refers to a filter A146 and a filter B148 at step 150, respectively, and judges the attribute of the contact pin instance which can be created from an available contact pin factory and available it. These attributes have the connection format and data format according to type of each pin factory of each filters 146 and 148, as mentioned above. Enquiry is held below using the set base enquiry mechanism explained in detail.

[0097] After finishing enquiry of the above-mentioned information, a third party control agent opts for the optimal connection format based on the range of the data format for which it referred before, and a connection format. This decision is made at step 152 and it can be made to do usage depending on which third party agents differed the same filter according to the requirement of the selected connection pass. A third party control agent uses a data crossover property (data intersection property), topology information, and a contact pin factory with both filters, and determines the best selection approach of a data format and connecting arrangement according to the actual filter graph created.

[0098] The input filter pin instance 154 uses the optimal detection information judged at step 152, and is created by the third party agent at step 156. Since the input pin instance 154 is a file object, a handle is returned from a creation process, but this can be used in order to send input/output request to the input instance 154. Furthermore, although creation of the input pin instance 154 is inspected in effectiveness, routing and the validation mechanism which were mentioned above with reference to drawing 4 - drawing 6 , drawing 7 , and drawing 8 are used

for this creation.

[0099] In order to complete connection, the output pin instance 158 is created at step 160 as a parameter in NtCreateFile Kohl using the handle of the input pin instance 154 created before. As effectiveness that the output pin instance 158 does in this way, and is created, internal IRP stack structure is created using a system file function manager and an input/output management function. Since it becomes possible to carry out consecutive processing of the original write-in command to the contact pin instance connected by various approaches in suitable sequence with a filter, the immediate-data flow between different filters is easy-ized by this stack structure. In order to perform this, it is required to create the input pin instance before the related output pin instance which supplies an input pin instance.

[0100] The stack depth parameter of a device object controls how many a stack location is created to IRP sent to this driver. Although it is assumed that the number of stack depth parameters is one when a device object is made for the first time, it is also possible to change later according to whether chaining of two or more drivers is carried out to one. In the present system, if required, this change will be made, when an output pin instance changes in "acquisition" or other condition from an initial "halt" condition. The state transition of a contact pin instance is a mechanism which determines the right stack depth parameter information for creating IRP correctly and processing it.

[0101] In order to assign correctly the contact pin instance set by which chaining was carried out, it is necessary to make it change so that it may come out of a contact pin instance from a idle state in specific sequence. that is, -- from the last input pin instance (this example input pin instance 154) -- starting -- output (for example, it connected) pin instance (this example output pin instance 158) of relation up to -- it is necessary to return to hard flow continuously If chaining of many filters is carried out to one, the input pin instance of the deepest filter or a bridge is the

initiation point of transition, and it must build continuously to hard flow until the initial output pin instance on a bridge or a filter is set up. the transition which in other words comes out of a idle state -- a chain -- ****** -- it is carried out like and must be made to have to obtain the stack size for which each contact pin instance is needed after a front contact pin instance As an example of representation, although it is not necessary to necessarily do so, and a contact pin instance changes from a idle state to an acquisition condition, the same purpose as the transition of the following explanation to which the transition to an acquisition condition comes out of a idle state about adjustment of a stack depth parameter for convenience is achieved.

[0102] After all pin instances are in an acquisition condition, stream read and writing can be taken out to a filter graph. In addition, the system which explains what should be observed here can make connection of a related input and an output pin instance in every sequence, and is having to perform only transition from a idle state first from a bottom up system, i.e., the deepest thing. Furthermore, reconstruction of a filter graph is attained so that it can change after initial creation. When a change is made, a state transition is performed only by the contact pin instance in a idle state, and right stack depth parameter information needs to be made to be acquired.

[0103] The contact pin factory found out on a filter expresses the location where a filter can consume and/or create data in a specific format. For example, a specific contact pin factory can support the data format from which some, such as a 16 bits 44kHz PCM audio and a 8-bit 22kHz PCM audio, differ. As mentioned above, about different functions, such as a contact pin factory and its data format, a reference can be made from a filter using a suitable property set mechanism and a system input/output function. An actual contact pin instance is created based on the information received from the pin factory.

[0104] If single stream writing or stream read operation is taken out by the user Mohd agent, in the streaming environment where consecutive processing of data is performed through the

connected filter, two Maine methods for IRP control can use it as a part of NETIBU function of NT operating system. individual by the 1st method -- IRP is created with each filter, is sent to the following filter and processed, and this filter creates new IRP, goes down a chain, and is processed one after another. By other methods, single IRP is used and it is delivered between continuous filters using the standard procedure prepared in order to interact with an I/O manager. For the interconnect sequence between filters not being important for when the 1st method which creates new IRP for every continuous filter on a chain is used, a filter is the destination (destination) of IRP. It is because it can act as Kohl of the I/O manager and he can be seen off in the filter of assignment of IRP that what is necessary is just to know. A thing important when the reuse of the IRP is carried out is that even the filter which created IRP for processing processes even the first filter which is performed so that it may begin from the last filter with which transition of the contact pin instance from a idle state receives Reuse IRP, and receives Reuse IRP to hard flow.

[0105] The gestalt and the example of mounting of this operation of an interconnect Carnell mode filter mitigate the complexity of driver development using IRP common use, make it possible to create a stronger driver, and have the advantage of increasing the efficiency of processing. The pin instance state-transition pass of a "bottom-up" guarantees that right stack sequence is created by IRP processed by the continuation driver, and has the stack depth parameter set with each suitable driver object. Furthermore, the present condition voice of a receiving-side input pin factory is checked in order to confirm whether the state-transition sequence was followed correctly. From such a reason, the communication link property of a specific contact pin factory judges the direction of a flow which may happen, and when distributing the state transition of a contact pin instance correctly, it supports it.

[0106] When creating an output pin instance (or IRP source), the reference to the file object

showing the input pin instance on another filter (or IRP sink (sink)) is handed over as some NtCreateFile Kohl. A suitable creation handler is performed as the multiplexing dispatch function, and the device object / file object hierarchy were used and mentioned above. This creation handler can access the device object of a filter (for example, filter B148 of drawing 9) with an input pin instance through an input contact pin instance file object (for example, input pin instance 154). A front stack depth parameter can be read and the stack depth parameter of the device object of a filter with an output pin instance is increased from a device object. For example, the device object relevant to a filter A146 has the stack depth parameter which is a device object relevant to a filter B148, and is increased in the connection shown in drawing 9 . Usually this is performed at the time of the transition which comes out of a idle state, and while a contact pin instance is in a idle state, routing of the IRP is not carried out.

[0107] When a filter processes IRP, it knows which stack frame or location should be accessed by using it with reference to the stack depth parameter of a related device object. [in an IRP stack including the information specified for / the / specific filters] Furthermore, the present filter prepares IRP for the next filters on a processing chain by reducing the stack depth parameter of a device object and positioning in the IRP stack location of the following filter.

[0108] It takes charge of a filter code preparing the next location in an IRP stack, and acting as Kohl of the I/O manager, and passing IRP to the following filter as specified. If it does in this way, it can specify whether which file object showing a specific contact pin instance receives IRP and associated data, and a filter processes. Therefore, standard-input/output manager Kohl like IoAttachDevice for carrying out the stack of each device object for sequential processing of IRP is not used.

[0109] What should be observed is not meaning creating connection between contact pin instances creating a device object new since the connection is expressed. The single device

object used as a foundation is used in order to support the instance of a filter, and all the contact pin instances on the filter. Although concrete information required for right data processing is saved in the context area of a file object and context information is left behind as it is, non-page memory usage is maintained at the minimum. Although what should furthermore be observed has explained the medium of the IRP base, it is being able to use other media for the communication link between interconnect filters. Direct function Kohl in non-host hardware and the communication link between hardware is one of such things.

[0110] Next, with reference to drawing 11 , drawing 12 , and drawing 13 , right creation of the software driver shown in drawing 1 (conventional technique) and drawing 2 (high level logic diagram of an interconnect Carnell Mohd driver), connection, and state-transition sequence are explained. Drawing 11 shows the processing step contained the logic structure enclosed with a box 162, and there. Drawing 12 contains the processing step enclosed with a box 164 in the flow chart which creates a contact pin instance, shows signs that interconnect of the Carnell mode filter is completed, and is shown in drawing 13 .

[0111] When it is in the condition of drawing 12 that all interconnect is performed, the Carnell mode filter system is in the preparatory state of R/W, in order to process. Input/output system uses the IRP stack information correctly set up according to the right state-transition process, and hands over stream read and writing to a different filter element through each contact pin instance. In addition, a certain kind of external software other than the agent used in order to create a graph offers the data of stream read and a light also including a bridge or the filter itself, and hardware.

[0112] After starting from step 168, the control agent 170 is step 180 and creates the instance of the reader filter 172, the decompressure filter 174, the effectiveness filter 176, and the sound rendering filter 178. Furthermore, connection between the reader filter 172 and a disk driver 182

is made, and data are carried in from a disk drive. Creation of each filter instance is performed by the user modal-control agent 170 by opening the file on the suitable device found out by the device I/O directory tree using standard-input/output Kohl. From this Kohl, the handle to the file object showing the instance of each filter is returned.

[0113] At step 184, a third party agent refers to the effectiveness filter 172, the decompressure filter 174, the effectiveness filter 176, and the sound rendering filter 178, and judges the function of a contact pin factory. These functions can create what kind of I/O pin instance, or a specific filter has a medium or a type of a data format and a communication path etc. which supports the instance of each contact pin factory how many, or is supported in each contact pin factory. Since it refers for these functions before using the property set mechanism explained in detail and the Carnell mode filter is supporting the suitable "set" (for example, property set), it is the requisite to follow architecture.

[0114] The connection pass by which chain NINGU was carried out creates a suitable contact pin instance, and such all enquiry information on step 184 is used in order to judge whether it is possible between each filter by connecting. A third party agent creates the filter graph which judges the type of a pin instance required for interconnect, and attains the given purpose.

[0115] The decision of the connection format based on the data format supported is made at step 186. If the topology information on a filter, a data format, and a data crossover property are used, the filter (hypothetical) graph on an assumption can be created. Since connection sequence is not important, it is not necessary to make it such but, and time amount can be saved when it is going to create a filter graph. If the filter graph on this assumption is created without an error, the relief that a third party agent can be trusted and can perform creation of the contact pin instance which interconnects will be obtained. When the filter graph on an assumption is created after creating this contact pin instance since an error is returned from a certain kind of enquiry if the

actual pin instance is not created, directions reliable in the ability to perform will be returned. It is possible to test, before the filter graph on an assumption interconnects also in this case.

[0116] If the right initial entry is known and it will be determined at step 186, an input pin instance is created, and it can interconnect and can express with the loop formation of the processing step surrounded with the box 164 of drawing 13 . This loop formation contains the processing step which begins from the input pin instance which is most separated from the source of a data stream in the distance. The input pin instance of this last is called a "deepest" pin instance, and this is created first and can continue the output pin instance related after it. Therefore, connection creates an output pin instance using the handle of the input pin instance created before.

[0117] This pattern is continued so that all input pin instances may continue after it and may be created before connection with a related output pin instance. Such a connection scenario is mere instantiation, connects each output and an input pin instance, and does not limit other possible approaches of forming connection between the Carnell mode filters by this invention. A filter can be connected in every sequence according to mounting, as long as the handle from an input pin instance is used for the creation period of an output pin instance when it connected on another filter. Furthermore, as mentioned above, it is possible to change into the filter graph after initial creation (and even in case of after use).

[0118] At the repeat of the beginning of a loop formation, the input pin instance 188 is created at step 190. If a handle is received from a creation function, the third party control agent 170 will create the output pin instance 192 at step 194 as a parameter by NtCreateFile Kohl using the handle. If this is performed by the first repeat, the sound rendering filter 178 will be effectually connected to the effectiveness filter 176 through the contact pin instances 188 and 192 which correspond, respectively. In the present mounting, "the lap (wrapped)" of NtCreateFile Kohl is

carried out as some function Kohl of API who can use a user Mohd client. Thereby, since a third party agent's user Mohd developer is released from the need of getting to know a detail, he can make single user Mohd API concentrate all related functions.

[0119] At step 196, a third party agent judges whether the input pin instance which should be created remains else. If it remains, an input pin instance must be created and the output pin instance to which it corresponds on another filter after it must be continued. Finally, all connection is made and the third party control agent 170 prepares a filter graph to stream-ized data processing.

[0120] It is created by the 2nd repeat of the loop formation by which the input pin instance 202 was surrounded as mentioned above with the box 164 by step 190, and another side and the output pin instance 204 use the handle of the input pin instance 202 as a part of the creation at step 194. Finally, in this specific example, by the repeat of the 3rd last, the input pin instance 206 is created and connection is completed by the output pin instance 208 following it.

[0121] The third party control agent 170 makes each contact pin instance change from a idle state to an acquisition condition in preparation for stream-ized data processing by the filter graph at step 197. in order to set up a stack depth parameter correctly in each of the device object of each filter, a state transition "goes back" the chain of an interconnect Carnell mode filter one by one until it begins it from "it being the deepest", or the last contact pin instance (for example, input pin instance of the last which receives the data for processing) and it reaches the first contact pin instance (for example, the first output pin instance which sends data into a graph) -- it is necessary to carry out like The first filter or bridge creates IRP, and a stack location is fully assigned so that IRP may receive in each kernel mode filter in a graph continuously efficiently and may be passed to it.

[0122] Finally, the third party control agent 170 takes out stream read and writing, and after he

processes data at step 198, he ends them at step 200.

[0123] As mentioned above, the handle of the file object showing the input pin instance connected there is required for each creation of an output pin instance. The creation handler of an output pin instance enables it to save "this refer to the file object" for access to the reference to the device object corresponding to an input pin instance of the present or the future.

[0124] If it explains more concretely, when carrying out the state transition of the stack depth parameter of the device object which manages an input pin instance from a idle state to acquisition or other conditions, thereby, it will become possible to access by the driver of an output pin instance. The value of the stack depth parameter relevant to an input pin instance is accessed, and it is increased and it is saved in the stack depth parameter of the device object corresponding to an output pin instance.

[0125] Although a stack depth parameter is used in order to judge where [of common IRP stack structure] the stack frame information for specific filters is put, these differs for every filter. If a filter is interconnected in this way and a state transition is performed by the right sequence, even if it does not make a communication link into user Mohd, Single IRP will be delivered and Lycium chinense will grow so that the chain of Carnell Mohd's interconnect filter may be gone down.

[0126] It is possible to have two or more instances on the basis of the same contact pin factory so that I may be understood from the above explanation. For example, an audio mixing filter can be processed, after mixing two or more pin instances and making it a single output pin instance. Each input instance is the same type and a filter can support only the input pin of one type. There is an example which carries out two or more inputs to one output as another example of such a configuration.

[0127] It is also possible to perform a reverse thing. That is, it is possible by a splitter filter's having a single input contact pin instance, and offering two or more output pin instances to

double a data stream. mounting actual from the connection mechanism mentioned above if this field becomes a familiarity person, so that I may be understood, and its requirements -- responding -- various -- voice -- like -- deforming -- various -- voice -- combining like is possible.

[0128] Since homogeneity and a standardization are attained when a driver developer makes all comp rye ANTO filters support the common mechanism (for example, a property set, a method set, and an event set) which can be mounted independently, a control agent can connect the comp rye ANTO filter offered by the different-species software provider with sufficient convenience. Furthermore, even if many of functions seen from a viewpoint of a contact pin factory are required of a certain situation, it may not be required of another situation. The decision of a contact pin instance needed is first made by the third party control agent who actually performs interconnect between different filters.

[0129] If this field becomes a familiarity person, an approach can still be variously incorporated as a computer program code means as computer instruction stored on computer read possible media, such as other media of this invention which are common in a magnetic disk, CD-ROM, and this field, and other underdeveloped common media, so that I may be understood. Furthermore, the important DS put on computer hardware memory can be created by the operation of the above computer program code means.

[0130] This invention can also be realized with other operation gestalten, unless it deviates from the pneuma of the basic feature of this invention. Although the gestalt of various operations mentioned above is as all points having explained, it is not limited to the gestalt of these operations. Therefore, the range of this invention is limited by only the matter which is not by the explanation mentioned above and is indicated by the claim. All modification belonging to the equal semantics and the equal range of a claim belongs to the range of this invention.

TECHNICAL FIELD

[Field of the Invention] The field of this invention is in the thing belonging to development of a computer software driver. This invention relates to the tool which offers the standardized access point, a software framework, a regulation, etc. while simplifying the code generation of a driver. the software driver to which this invention was more specifically written by different developer -- Carnell Mohd -- interconnecting -- user Mohd from Carnell Mohd -- or it is related with the standardized method which makes consecutive processing possible between different drivers, a computer program product, and DS, without performing inefficient transition to Carnell Mohd from user Mohd.

PRIOR ART

[Description of the Prior Art] Usually it is made so that a software function may be offered, and a software driver mitigates a system overhead, lessens constraint relatively, and is performed under the operating system so that hardware may be controlled. If it is made such, since the system code performed in order to guarantee normal actuation (behavior) and to carry out "the trap (trap)" of the interference with another process under activation under invalid access or an operating system will decrease, it has accelerated that a driver accesses hardware and serves a time critical processing demand (time critical processing request).

[0003] the level of operation or "Mohd (mode)" from whom the operating system has responded to the amount of access, and the security function mounted (mounting), and shoes differ -- **** -- it usually comes out that it is. For example, the usual application program was performed with the minimum priority, and has equipped the proper place fully with the security device for preventing interference with other applications. Only by hardware's letting the interface put under control

pass, it is accessed. Although it is made for convenience the thing of explanation for which this is named "user Mohd (user mode)" generically, the Windows NT operating system currently used as a part of example of mounting of this invention which lower-** is supporting user Mohd. Similarly, the operating system of other most has a mode of operation equivalent to "user Mohd" regardless of the complexity.

[0004] On the other hand, the execution priority of a driver is far high, and since security protection is performed by Mohd of a low operating system, it is possible to access a driver to the actual hardware which a driver deals with directly in many cases. Much application is made into this thing [this being called "Carnell Mohd (kernel mode)" generally, and calling it such also on these specifications with the Windows NT vocabulary,], although it is more loose, and an advantage will be acquired if it is made to perform by performance-oriented Mohd more. Also in this case, other strong (robust) operating systems have equivalent Mohd functionally.

[0005] Since it has the intention of the general conception of a software driver controlling specific hardware, a driver is separated from mutual, and is developed and, usually it is provided by the hardware manufacturer. The metaphor does not need to know the existence, if the software driver relevant to an add-in hardware card (add-in hardware card) which offers a certain kind of I/O service through the definition of a device does not need to communicate with all other drivers.

[0006] Since processing with desirable making it operate by Carnell Mohd operates [lack / to which the function which can communicate with such exclusive-like approach of development of a driver between drivers according to a situation relates / make / it / to make] by user Mohd, it happens to become disadvantageous in respect of performance.

[0007] As one of the typical examples of the present program which cannot use easily the Carnell Mohd driver used in this specification, a different processing block with which a user is called a filter is chosen, these are combined with one, and there is a graphing function to enable

it to operate the stream of multimedia data continuously. data are a series of samples showing a sound or video -- typical -- a processing block -- elongation (decompression) of compressed data Processing and special effect (special effects) Processing and CODEC Rendering block (rendering block) which changes a function and data into an analog signal etc. -- it is.

[0008] The graphing functional division of a program interconnects, the operation is controlled, and the above filters are a user's input and relocation (rearrangement) of a processing block. It is typical to be placed by user Mohd so that it may respond. There is coordination in the stream of multimedia data, and since a lot of data are generated, performance has been an important problem. In general-purpose operating system, since system performance falls as a result of carrying out by repeating delivery and a change between user Mohd and Carnell Mohd and inspecting implicitly whether such transition (transition) is effective on security, a certain kind of multimedia application can be used.

[0009] Furthermore, since the processing block is connected to much time amount and hardware, transition between components, which are user Mohd and Carnell Mohd, will be performed numbers of degrees. When such transition is performed, the overhead of another gestalt to which the total performance of a multimedia processing system is reduced will be included. If it changes between user Mohd and Carnell Mohd, the overhead related when copying data between different buffers arises, and this should not be produced if processing is left Carnell Mohd.

[0010] Another fault which happens when performing transition to user Mohd from Carnell Mohd is that the approach of carrying out scheduling of the activity task with an operating system is restricted. If it is possible to do an activity Carnell Mohd, since the twist performance-oriented scheduling approaches, such as a software interrupt and deferment procedure Kohl (deferred procedure call-DPC), can be used, system performance is optimized further.

[0011] Furthermore, it may be convenient if a different driver developer enables it to create a mutually connectable driver according to a common interconnect method and a common definition. If it does in this way, since every driver ability written to the common interface can interconnect in the system of a functional processing block, all data transition will be performed by Carnell Mohd. furthermore, the driver developer from whom many differ when well-known specification is used -- mutual -- it is operational and it becomes possible to develop the driver software which can interconnect.

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] The purpose of this invention is improving system performance by interconnecting by the method which had the software driver standardized, and preventing the Mohd transition of an operating system at a data-processing period.

[0013] Furthermore, the purpose of this invention is offering the extensible base mechanism for software drivers which can be interconnected by the third party developer.

[0014] Another purpose of this invention is making it generate more performance critical processings in Carnell Mohd.

[0015] Furthermore, another purpose of this invention is that a third party component enables it to interconnect a software driver.

[0016] Although other purposes and advantages of this invention are indicated by the following explanation, the part has what is understood from the explanation, and the thing which can carry out learning by carrying out this invention. the instrumental the purpose and advantage of this invention are separately indicated to be by the claim -- the means and combination of RYUMENTO (instrument) realize and it is obtained.

MEANS

[Means for Solving the Problem] In order to attain the above-mentioned purpose, it is embodied below, and according to this invention currently explained to the wide sense, the approach and computer program product for interconnecting a software driver by Carnell Mohd are offered. Even if it does not depend for a certain given driver or filter on user Mohd's agent by supporting and defining the connection "a pin factory (pin factory)" which creates other pin instances (pin instance) put on other drivers, and the pin instance of a certain kind of type which can interconnect, consecutive processing of the processing message is made to be carried out by Carnell Mohd by the driver. If it does in this way, the flow of the data is completely carried out by Carnell Mohd, and since the overhead of crossing user Mohd becomes unnecessary, data will be processed efficiently.

[0018] The third party agent who wishes to connect a comp rye ANTO driver (compliant driver) refers for whether the function is in a driver. As such a function, he is Ince Tan Scheidt (instantiate) about a contact pin instance. It may say what kind of contact pin factory can be used for carrying out, and the related characteristic of the property of the input of the type of the data processed, a data format, a transfer rate, a transfer medium or Mohd, and a contact pin instance or an output is included in this.

[0019] After a third party agent with typical performing by user Mohd makes a reference about

the function of one or two comp rye ANTO drivers or more, the agent does "chaining" of two or more drivers to one, and determines the best connection characteristics by which data are processed the optimal between drivers. This decision step finishes enquiry of all driver ability, and after being able to choose the optimal traffic engineering standard, it is performed.

[0020] Next, a third party agent interconnects a driver by creating the instance of the required contact pin on each driver using a pin factory. An agent specifies a data format and a connection format as a part of contact pin instance creation. Furthermore, the reference to the contact pin instance created before is specified in the demand which creates other contact pin instances, and enables it to make connection between contact pin instances.

[0021] An actual contact pin instance is created with the gestalt of the instantiation-operation mounted under NT operating system by the creation I/O statement which returns the handle (handle) to a "file." This creation input/output request has included the driver instance handle to DS and reference which show a data format and connection format of a contact pin instance.

[0022] In order to create a comp rye ANTO driver, the driver developer is supporting a certain kind of standard function to which user Mohd's agent refers about a function, and enables it to perform interconnect between drivers. This is attained by use of "the set (set)" (that is, a property, a method, and the set of an event) which mounts the function to need with the gestalt of a certain operation currently built on the Windows NT operating system.

[0023] The set is logically defined as a thing with GUID (globally unique identifier) (globally unique identifier) which specifies the whole set, and RUID (relatively unique identifier) (it is a unique identifier, for example, an identifier relative within the set itself, relatively) of each functional element in a set. Each set is related only with one or two IOCTLs(es) (IO Controls) (I/O control), and IOCTL is combined with the specification of a set and controls all interaction with a driver.

[0024] The set of three types is used with the gestalt of this operation. That is, they are a property set, a method set, and an event set. it is used in order that a property set may manage a value or the set points, such as sound volume and a transfer rate, within a driver, and Kohl is going to acquire the property value -- and/or, the flag which shows whether it is carrying out [which will set up a property value], and IOCTL single together are used. A method set is used in order to manage the operation which can perform a driver like allocation of memory, and Flushing (flushing) of a buffer, and it acts as Kohl of the specific method using single IOCTL. An event set is a device change notice and data starvation (data starvation). Like a notice, it is used in order to manage the event relevant to processing of a driver, and two IOCTL(s) are used. One is for making a specific event enable (enable), and another is a disable (disable) about a specific event. It is for carrying out.

[0025] In order to use a set, input/output control operation is started using the required data of the reference and others to DS with specific IOCTL, GUID of a set, and RUID. For example, when setting up a volume property by the sound card driver, IOCTL of a set property will be used, suitable GUID of the property set with the volume set point will be specified, it will direct that the pinpointing RUID in the set shows the volume property, and input/output control operation including the new volume set point will be performed.

[0026] In order to make a reference about the set supported, Nur GUID is used together with an enquiry flag by the specific set type specification IOCTL (for example, the property set IOCTL, the method set IOCTL, or event enabling (IOCTL)), and the list of sets GUID supported is returned. In order to make a reference about the property with which it was supported in a certain set, a method, or an event, Set GUID, the set type IOCTL, and an enquiry flag are used together with the operation which returns the list of supported RUID.

[0027] Although the minimum function can be mounted in order to support a comp rye ANTO

driver if a general-purpose set mechanism is used, expandability is still unrestricted. As long as the specific set is mounted, mutual actuation is possible for a set, and in order to create the system of the driver which can interconnect, it can be defined as the specification which can be coded independently by the driver developer from whom a large number differ. Furthermore, the property of the option which it not only can define an indispensable property with the need of supporting, a method, and an event, but can be mounted depending on the function and extension of a driver, a method, and an event can also be defined as this specification. It is also possible to incorporate an additional function by defining the set with an original driver developer other than the minimum fundamental similarity demanded, and assigning GUID to the set. If it enables it to enumerate the functions (that is, to hold enquiry of GUID supported and RUID) supported, a call side (caller), such as a third party control agent, can also perform suitable compensation depending on the function of a filter in which it also becomes a foundation to adjust expectation.

[0028] Although the above of this invention, and the other purposes and descriptions are as being indicated in detail by the following explanation and the claim, it is also possible to carry out learning by carrying out this invention explained below.

[0029] In order to show the approach how the above of this invention, and the other advantages and purposes are gained, below with reference to the gestalt of the concrete operation currently illustrated by the accompanying drawing, this invention explained briefly above will be explained in detail. An accompanying drawing is made to describe this invention more concretely and in detail, and to explain it while if the gestalt of typical operation of this invention was shown, it is not alike and ****, therefore the range of this invention are not limited uses an accompanying drawing hereafter under comprehension.

[0030]

[Embodiment of the Invention] The vocabulary "user Mohd (user mode)" used on these specifications is the level of operation in an operating system in case the great portion of program written by the user is performed. User Mohd's level of operation has the highest security level, and in order to prevent interfering in a certain application program, application program with an another process, or a process, it is typical that a lot of overheads are consumed. Furthermore, access to a system resource is strictly controlled through a specific interface, and though an execution priority is not the minimum, it is general [the execution priority] that it is one of the minimum priorities.

[0031] The vocabulary "Carnell Mohd (kernel mode)" used on these specifications is the level of operation in an operating system with very less constraint than user Mohd's mode of operation. As Carnell Mohd's program, or an example of a process, it is the software driver (software driver) which controls a hardware component. It contains. As an example of representation, Carnell Mohd's program tended to be influenced by performance, therefore the overhead on actuation has become less than user Mohd's program. Furthermore, access to the system resource of hardware or many was not restrained, or its constraint has become less than the case of user Mohd's program far. The program code performed by Carnell Mohd trusts the order rightness of a programmer and conformity to a regulation (convention), and it is made for a system to operate good in many cases (for example, it is made not to destroy the address space of another program). as another vocabulary showing Carnell Mohd -- "-- a truss -- TEDDO (trusted) -- there is a" code.

[0032] The vocabulary "the driver (driver)" used on these specifications is a software driver program with typical performing by Carnell Mohd. Vocabulary called a driver may mean the program part which divides and gives the actual executable program loaded on an operating system, or a function of a certain kind. In many cases, the driver is related with the hardware of a

certain gestalt, but it comes out so and a certain need is not not necessarily.

[0033] Then, it is the thing of the functional division placed into the software driver, and in this, the driver itself is contained, and the vocabulary "the filter (filter)" used on these specifications is exhibited, in order that a connection point may transmit data through a filter. For example, the software driver may have one single function, if the filter with which some differ may be supported. Furthermore, it may connect with one internally and two or more filters from a different driver which opens the connection point to I/O externally may be collectively referred to as a single filter. moreover, vocabulary called a filter at more generic semantics -- elongation (decompression) etc. -- it may mean the operation performed like regardless of whether it happens with the software driver filter performed by Carnell Mohd, or it happens in another program code performed by user Mohd

[0034] The vocabulary "the driver object (driver object)" used on these specifications is the entity of the operating system defined by the operating system which manages a software driver and tells this as a system resource.

[0035] The vocabulary "the device object (device object)" used on these specifications is the entity of the system level defined by the system, and it defines whether this tells a part of function of the driver which can be used as a system resource, and can use the function and alien-system component of a driver. Both of driver objects and device objects are typically created at the time of loading of driver software, and initialization.

[0036] The vocabulary "the file object (file object)" used on these specifications is an entity of an operating system which manages the call of the resource specified by the device object, and this is defined by the operating system. A file object offers the context about the operating condition of a driver object. Furthermore, a file object can be connected with another file object and a hierarchy target, if the former file object is specified as "parents" by the creation time of a new file

object. A file object is used for management of all the I/O statements typically operated on a data stream.

[0037] The vocabulary "the data (data)" used on these specifications is all information processed through the filter of Carnell Mohd who interconnected. Although such data contain the media data showing video, an audio, a text, MIDI, etc., in the case of other applications, control information and a parameter may also be included. For example, although the Carnell mode filter graph may be used by process control operation, the control information delivered between different filters is used for the ******** sake there in the control signal which actuates machines. Although the example of a media processing system is shown, other applications can acquire an advantage from the system of the interconnect Carnell mode filter currently explained to this specification by the same approach.

[0038] Explanation of this invention is indicated by this specification within the context of the Windows NT (trademark) operating system currently offered from Microsoft (trademark). Furthermore, in order to understand the gestalt of the suitable operation currently explained to this specification, the knowledge of the I/O architecture of Windows NT is the requisite. input/output system -- and -- NT As a suitable document which explained the operating system at large, although there is Helen Custer work "the interior of Windows NT (Inside Windows NT)" (Microsoft Press issue), this book constitutes some of these specifications by citation.

[0039] Although it is explained by the following explanation of system entities, such as a driver and a file object, a device object, and a driver object, what kind of how to work these adopt on a Windows NT operating system, this invention can be mounted on other operating systems with a similar component, and is unrelated to whether these operating systems are using the same vocabulary so that I may be understood, if this field becomes a familiarity person. For example, if another operating system has the entity which operates as a file object, the entity can be

interpreted as a file object regardless of the actual name.

[0040] First, if it explains with reference to drawing 1 , the example of a system of illustration reads the stream of sound data in a disk drive, and is a rendering (rendering) about the sound data. It carries out and is made audible from a loudspeaker according to the conventional model. Although an certain amount of data is stored in the hard drive 20, this expresses the sound with the gestalt of the digitized sound sample. Other sources well-known in the digitization information sent as the source of a sound data stream using the telephone line, the digitization information from a network or other communication link packets, and this field otherwise can be considered. A data stream consists of digitization samples and time interval information is associated by the explicit time stump information by which these samples are added to a data format, a regulation, or each sample. Carnell Mohd's disk driver 22 interacts with the disk drive hardware 20, and is put under control of user Mohd's reader (reader) program component 24. a control agent (controlling agent) -- although a component which is different in order that 26 may perform the rendering of sound data is managed -- dynamic graphing function (dynamic graph building capability) When it has, a different software component is assigned dynamically and enables it to offer custom-made filtering or other processing paths according to assignment of an end user.

[0041] The reader component 24 interacts with a disk driver 22 using the standard-input/output control interface of an operating system, reads compression sound data in a disk drive 20, and is the address space (address space) of a user Mohd process. It serves to put into the buffer assigned by user Mohd as a part. Next, the decompressure (decompressor) component 28 elongates compressed data to the elongation (decompressed) format suitable for processing. Like illustration, this whole step is performed by user Mohd using the process actuation (processbehavior) insurance mechanism of an attendant low priority.

[0042] the effectiveness filter (effects filter) 30 -- data -- actuation -- in addition, the effectiveness

filter 32 of the attachment which a certain kind of special effect is acquired, and operates by Carnell Mohd -- **** -- it is. Furthermore, if the effectiveness processor 34 may exist, it may operate by the software by which the whole effectiveness filter emulates an actual hardware processor. In order to access the effectiveness filter 32, the effectiveness component 30 uses a system input/output control mechanism, and transmits data and control to an effectiveness filter. Also in this case, Carnell Mohd / user Mohd boundary is crossed, and this transition is performed.

[0043] The effectiveness filter 32 controls the effectiveness processor 34, and the need or the special effect made desirable is made on the data. This copies all data from the effectiveness component 30, and is performed by moving the copy to the effectiveness filter 32 again also at the effectiveness processor 34 depending on an actual system configuration. Although, as for many software effectiveness components, the hardware processor is associated, other components are functioning completely within the system software performed on a host processor.

[0044] If control and data are returned to user Mohd at the time of completion of processing of the effectiveness component 30, this will be transmitted to the sound rendering component 36 next. The sound rendering component 36 transmits control and data to the sound rendering driver 38, and in response, a sound rendering driver is processed and it controls a sound card 40 to carry out the rendering of the filtered data as a sound from a loudspeaker 42. As mentioned above, in order to recognize variety existence, the transfer between user Mohd and Carnell Mohd has made the rendering of sound data inefficient, so that I may be understood easily. Like a continuous sound or a video stream, if it lessens copying data numbers of degrees between different buffers while multimedia data lessen transition of these inefficient nature and control for the property in which it is tended to influence timing, they are convenient.

[0045] The gestalt of 1 operation of this invention used on these specifications consists of

services offered on Windows NT operating system architecture. This service is divided into the software component which the user of a system accesses and from which some differ. The 1st is user Mohd's API and this contains the routine for creating a contact pin instance and other file objects showing specific functions, such as a clock mechanism and a buffer allocation mechanism. Others are equipped fully with the routine and DS which support creating the driver to which a driver developer follows standardization architecture as a more important thing. If these functions currently prepared for the system are used, a different driver developer can create the comp rye ANTO driver which interacts mutually based on specific architecture. A user Mohd agent communicates with a comp rye ANTO driver through the environmental subsystem under activation by user Mohd who communicates with the system service of NT executive and an I/O manager. This is the same standard-input/output mechanism over other the I/O of all, and uses the existing system service as much as possible in this mounting of the gestalt of suitable operation.

[0046] The architecture of the system of drawing 1 using this invention is shown in drawing 2 . The control agent 44 refers to the driver told, interconnects according to a data format and a connection format, and is made to perform a rendering by Carnell Mohd completely. Furthermore, since a control agent receives the notice of an important event, he can control if needed. Termination of processing, the data starvation situation, etc. are included as an example of such an event.

[0047] With this configuration, sound data are read in a disk drive 46 by the disk driver 48, as mentioned above. The reader driver 50 controls a disk driver 48, and is related with the disk driver 48 and the "perpendicular direction" according to NT layer (NT layered) I/O architecture using the former like the direction. The vocabulary of a "perpendicular direction" and a "horizontal direction" is used in order to distinguish the driver connection (perpendicular direction)

made as a part of NT layer I/O architecture now, and the connection (horizontal) according to the interconnect Carnell Mohd driver dynamically performed by the third party control agent.

[0048] According to the connection method explained below, the decompressure driver 52 interconnects "horizontally" and the reader driver 50 is managed by the control agent 44. Decompressure 52 hands over data and control in the effectiveness filter 54, after developing by Carnell Mohd. An effectiveness filter hands over data and control to the sound rendering driver 58, after applying special effect using the effectiveness processor 56 if needed, and this driver controls a sound card and carries out the rendering of the data as a sound from a loudspeaker 62. Since the amount of overheads which usually happens will decrease if two or more transition between user Mohd and Carnell Mohd will be lost if processing is left Carnell Mohd and it processes by user Mohd so that I may be understood from drawing 2 , the advantage of an effectiveness side is acquired.

[0049] Next, it explains with reference to drawing 3 . Drawing 3 is the logic diagram showing the hierarchical property of the system object related to the interconnect software driver according to the gestalt of 1 operation of this invention. Since the software code image when being loaded to memory which can be performed is expressed, the driver object 64 is created. The driver code image includes the whole function of a driver, and the driver object 64 includes the information about images, such as a class of the location put on the system, and driver supported.

[0050] According to each type of the function which can be independently accessed by the control agent, it is device object 66a. -66N Created by I/O directory structure, these objects are available and express a different function accessed by user Mohd's client. As for these, it is typical to express a filter or other functional divisions which can be used independently. The driver object 64 and device object 66a -66N As the enclosure box 68 shows, it is created at the time of install of a driver code.

[0051] Historically, a device object exists for every element of physical hardware. However, the flexibility of the newest input/output system enables a device object to express the filter completely mounted by software. Therefore, it is easy to create a device object for every instance of the filter chiefly mounted by software. Therefore, it is also possible to also mount so that, as for a software filter, each instance expressed with the device object may have the correspondence relation of 1 to 1 with a device object, and to mount so that the multiple-files object to which each expresses the client instance of a filter may be managed according to technique with a single more traditional device object.

[0052] At a device object and the example of illustration, it is device object 66a. Above, a file object is created and this expresses the independent instance of a function expressed with the device object. A device object expresses a filter and the file object expresses the actual instance of the filter used by the specific entity to managing two or more instances of the filter. Therefore, the file object 70 is device object 66a. It is the instance of the defined filter.

[0053] In order to use a filter, a control agent or other user Mohd clients open a file on a device available within I/O directory structure. The file object which has stored suitable context information is created, and the handle to the file is returned to a user Mohd client. He is the twin [as / whose all of a file object are the children of the same device object although a file object can be connected hierarchical by specifying a "parent" file object as creation time] (sibling). It also has relation.

[0054] The context information in a file object consists of "a condition (state) etc." of an entity etc. which the information which manages an input/output interface with a user Mohd client, and a file object express. There is information which a system needs for context information, and custom [which can give still more nearly special semantics] possible area is also included. The example of the direction is shown in the explanation part of validation (validation) and the in PURIME

tension of an IRP routing (routing) method which lower-** using custom possible area.

[0055] In order to offer a contact pin instance, the file object 70 showing a filter instance is used as parents, and the child file object showing the contact pin instance of a specific filter is created. To referring for the file object 70 about a definition and availability of a contact pin factory, an actual file object uses a specific file object as a suitable information context, it is created for every instance of a pin factory, and a contact pin instance is created effectively and correctly. For example, the file objects 72 and 74 express the contact pin instance of a filter expressed with the file object 70, and are connected with the file object 70 hierarchical. After it goes into a filter instance (expressed with the file object 70), the contact pin instance expressed with the file objects 72 and 74, respectively can be made the data pass which comes out from there, and it can use this in order to connect with other contact pin instances, when forming a series of chain filter or other driver ability.

[0056] If the file object of others [completely] similarly is connected with a pin instance and a hierarchy target with it being expressed with another file object to which a pin instance expresses a filter instance, and a file object with hierarchical relationship, and offering the context information on a pin instance and other functions are expressed, right context information will come to be acquired. Context information is required in order to distinguish a certain pin instance from other pin instances like a pin data format and a communication link type according to each parameter used for creation time.

[0057] Like a buffer allocation mechanism and a timing mechanism, an individual context or other actuation entities which require one of user Mohd's control through a handle can be expressed with a file object. Furthermore, the hierarchical relationship between file objects (for example, buffer allocation mechanism related with the specific contact pin instance) is establishable by specifying a father-file object as the creation time of a child file object, if required.

These parentages exist in order to determine the relation and structure between the file objects showing an actuation entity. Furthermore, since the "parent" file object of a particular type can make only a type of a certain kind of "child" file object, a creation validation mechanism is needed so that it may explain below. As for such a file object, also in this case, the corresponding handle has become available at user Mohd, and these handles are returned to a client through system API Kohl, such as NtCreateFile.

[0058] The handle to a file object is used by user Mohd clients, such as a control agent, in order to communicate with the Carnell Mohd driver. The hierarchical chain of a file object, a device object, and a driver object is an entry point (entry point) with which a I / O subsystem goes into return and an actual driver code to a driver object through a file object and a device object with hierarchical relationship. It enables it to reach. Such an entry point is the reference (for example, pointer) which points out the function in a software driver code. Furthermore, object path between a specific file object and a driver object with the entry point to a software driver code (pathway) Each of the object which is upwards also offers the reference to the DS used when carrying out routing of the IRP correctly according to routing and the validation mechanism which lower-** besides context information important when a I / O subsystem creates IRP.

[0059] The handle to a file object and an alien-system object is only for processes, and serves as a means when communicating with the object from which a user Mohd process serves as a foundation. What should be observed is being able to create two or more handles, in order to refer to the single system object used as foundations, such as a file object. It means that this can supply information to the pin instance to which two or more applications were expressed with the file object.

[0060] As one of the information elements which become important when a different driver is interconnected, it is the depth (depth) of a device object stack. There is a parameter. This shows

the IRP stack location of a specific driver object. If it does in this way, since an I/O manager is used and can be delivered between the drivers which interconnected using single IRP, IRP will be created separately, and performance improvement can be offered rather than it transmits it among various interconnect drivers. It is possible for each driver to let as an option Kohl, a suitable I/O manager, pass, to create new IRP for every continuation communication link, and to also make each new IRP transmit to the next driver on the chain of the driver which interconnected.

[0061] Next, it explains with reference to drawing 4 - drawing 6 . Drawing shows the extension of the system driver object which makes possible validation of file object creation of a different type, and routing of the input/output request packet (I/ORequest Packet:IRP) to a suitable handler, a device object, and a file object. Drawing 4 shows the driver object 76 showing the executable code which mounts one, or two a filter or other driver ability or more. Within a driver object, Windows NT architecture is demanding the reference to the creation handler which the software driver developer prepared. According to the gestalt of this operation, the multiplexing dispatch function (multiplexing dispatch function) 78 is referred to from the driver object 76 as a creation handler, and it is used in order to carry out routing to a specific creation handler according to the type of the file object which has a message created. The operation of the multiplexing dispatch function 78 is explained below with reference to the flow chart shown in drawing 8 .

[0062] It is possible to make these the same function according to how similarly other handlers of a driver object are mounted by showing a multiplexing dispatch function. In other words, referring to [of each type] the I/O handler (for example, reading writing, device control, etc.) used the context information in the extension data in a device object, and a file object, and have pointed out the multiplexing dispatch function which carries out routing of the message to a suitable handler so that it may explain in detail below. By creation operation, the extension data in the

device object which refers to a validation table are used, when there is no assignment of a father-file object. If there is assignment, the context information on a father-file object shows the right validation table.

[0063] By the driver developer, drawing 5 can be used by request and shows the driver object 80 with the specific device extension area 82 including the information only for drivers. It is called the file type validation table 84 to the location as which it defined in the device extension area 82 of the driver object 80, and the reference to DS including the string expression of the file object type 86 and the reference to the creation handler 88 by which each file type exception expressed is related are put on it. A creation multiplexing dispatch function uses the file type validation table 84, inspects the file object type created, and hands over control to a suitable creation handler after it. The explanation part of the following drawing 8 explains this in detail. The string inspected is ******** which is found out by IRP creation demand and used with user Mohd's NtCreateFile function Kohl from a file name string. NtCreateFile function Kohl is performed in a user Mohd function cel, in order to create a pin instance or other mechanisms.

[0064] Drawing 6 shows the file object 90 with the file context area 92 released in order that a software driver developer may use it. Reference is performed from the file context area 92 to the IRP demand handler table 94. The type with which the IRP demands 96 differ is related with the specific handler 98, and a suitable multiplexing dispatch function accesses a right handler using this information. In determining a right creation handler, the DS called the file type validation table 100 was referred to, and the reference 104 to the creation handler by which each file type exception expressed as the string expression of the file object type 102 is related is contained there. In the case of the child file object (that is, file object which has not a device object but another file object as parents), the type is expressed with the string in the file object type 102, and the string compared. If a match is found, a related creation handler will be accessed using

the reference related with the file object type string who was in agreement among reference 104. If a match is not found, since the demand is invalid, an error message will be taken out.

[0065] Next, if it explains with reference to drawing 7 , drawing shows the installation procedure for setting up creation validation and a mechanism. At step 106, an installation program creates the reference to a suitable multiplexing dispatch function in a driver object. As shown in drawing 4 , the creation handler has pointed out the general-purpose multiplexing dispatch function. Similarly, all referring to [other] the handler in the driver object 76 are other general purposes (generic) which have close relation with a specific handler. The multiplexing dispatch function is pointed out if needed. As an option, each referring to the handler can also point out the same multiplexing dispatch function, and in that case, this dispatch function can carry out routing of it to a suitable handler, after processing an IRP demand. Since the multiplexing function by this approach needs to receive demands (for example, creation, writing, etc.) of a different class, complicating is not avoided.

[0066] Next, each device object created as a part of install of a software driver executable code at step 108 is adjusted so that the file type validation table 84 may be referred to, as shown in drawing 5 . Processing of an IRP demand is started from a multiplexing dispatch function at step 110 by the last using the file type validation table 84 referred to from the suitable device object 80.

[0067] If a file object is created, the suitable IRP dispatch table 94 is created, and if required, it will be referred to together with the file object type validation table 100 by which the index was carried out. Creation of a file object type validation table is performed within the creation handler prepared according to the file object type. DS is created, this expresses the IRP dispatch table 94 and the file object type validation table 100, and the reference which points it out is stored in a specific location together with the file context information 92 on the specific file object 90 created.

[0068] Next, if it explains with reference to drawing 8 , drawing is a flow chart which shows the

operation and its validation mechanism of a creation multiplexing dispatch function, and the interaction with the DS referred to from a system driver object, a device object, and a file object is also shown there. At step 112, a user Mohd process transmits the input/output request which creates a file object. This I/O creation demand is performed by calling the system API of NtCreateFile. At step 114, an I/O manager transmits IRP to the multiplexing dispatch function 78 based on the reference in the driver object 76 (refer to drawing 4).

[0069] If the multiplexing dispatch function 78 receives IRP of a creation demand, a test will be performed at step 116 and it will be judged whether there is any father-file object. Although information required in order to make this judgment is in the IRP itself, this is originally prepared by the user Mohd process. A user Mohd process prepares the handle which refers to a "parent" file object as a part of creation demand, and NT system creates IRP with refer to the right to a "parent" file object.

[0070] If there is no father-file object, the branch to the right is taken, and the multiplexing dispatch function 78 will use the device extension 82 from the suitable device object 80, and refer to the file type validation table 84 for it at step 118 (refer to drawing 5). The multiplexing dispatch function 78 inspects a file object type at step 120 by using the validation table 84 by comparing the string in a demand with the string of the file object type 86.

[0071] If it is judged that the string is in agreement at step 122, a suitable creation handler will be accessed at step 124. If not in agreement, a creation demand is refused at step 126. The creation handler accessed at step 124 creates a file object at step 126, or is made to create it. Using the created file object, a suitable creation handler creates "refer to [which points out the IRP dispatch table 94 which was being created before] the file object" in the file context 92.

[0072] It is judged whether it returns to step 116 again and a father-file object exists. If a father-file object exists and it is [it was judged at step 116 and] contained in IRP relevant to a

creation demand, the multiplexing dispatch function 78 will use the file context 92 from the father-file object 90, and refer to the IRP dispatch table 92 for it at step 130 (drawing 6). In a creation demand, refer to the file type validation table 100 for the multiplexing dispatch function 78 at step 132. Using the file type validation table 100, the multiplexing dispatch function 78 inspects a file object type at step 133 like the above by comparing the string in a demand with the string of the file object type 102.

[0073] If the string is in agreement and it will be judged at step 134, a suitable creation handler will be accessed at step 138. If not in agreement, a creation demand is refused at step 136. A suitable creation handler is used, a file object is created by 140, and a creation handler creates the reference which points out the IRP dispatch table 94 which created the new IRP dispatch table 94 of the file object created newly, and was created newly at step 142 in the file context area 92 of the file object 90 created newly. What should be careful of is that the same file object structure shown in drawing 6 in both cases is used, in order to explain an interaction with a father-file object and the child file object created effectively. Although the same structure exists in both cases (after a new file object was created), the information in which the usage differs and contains these also differs.

[0074] If a contact pin instance is created, contact pin ID is handed over and this shows the pin factory in the file "supports" creation of a pin instance always. Contact pin ID is possible also for inspecting as a string on a validation table completely the same with a file object being inspected, and that mounting approach is also various so that I may be understood, if this field becomes a familiarity person.

[0075] In order to make connection between different drivers, the common mechanism for confirming that a certain driver is supporting such interconnect is required. This common mechanism must make it possible to clarify a filtering function including a contact pin factory

function. Furthermore, this kind of mechanism also needs a thing extensible so that a driver developer's flexibility may be improved.

[0076] A comp rye ANTO driver is defined, and in order to make it possible to clarify a function, one mechanism chosen with the gestalt of this operation is named the "set" of a related item. When this is used together with the existing I/O communication link mechanism, it is a convenient mechanism. The set is logically defined as a thing with GUID (globally unique identifier) which specifies the whole set, and RUID of each functional element in a set, relatively a unique identifier, for example, the relative identifier in the set itself. The DS of in order to carry out operation together with a set identifier and the selected RUID item, an and also [it is the need] is handed over as some input/output control Kohl as a parameter using a filter handle. In order to mount the perfect system of a function, it is enough just to assign a small number of IOCTL. Since the set of three different classes is established according to the function when mounted, needed IOCTL is a total of four pieces. How to use a set may differ in other mounting. Specific IOCTL notifies the selected element (RUID is used) to the handler of input/output control that it is interpreted or used by a certain approach. Furthermore, a control flag can be passed together with GUID and RUID, and control information can be specified in more detail.

[0077] The first set type is a property set and this is used together with the value or the set point placed in a driver and on related hardware. As an example of such the set point, there are a transfer rate, volume level (sound volume), etc. One IOCTL is related with a property set and a control flag is "get". A property command and "set" The property command is distinguished. If it does in this way, it can be used so that the same DS may set up or acquire a specific property, and a driver can be determined based on IOCTL which had required action used. A right property is specified by the set identifier (set identifier) which consists of combination of unique GUID and RUID.

[0078] A method set is another set type used, and this is the set of action which can be performed by the driver. IOCTL required since a method set is specified is only one, and the right method to actuate is specified with the combination of unique GUID of a set identifier, and RUID. The method includes the function in which initialization for being used in order to control a driver, and using a driver, and a buffer are clear.

[0079] An event set is used in order to manage the event relevant to driver processing of a device change notice, the notice of data starvation, etc., and other notices defined by the convenient set when it was used with user Mohd application. Two IOCTL(s) are used, one is for enabling a specific event and another is for disabling a specific event. The DS required for the given event identified by RUID can be shared regardless of whether an event is enabled or it disables.

[0080] In order to use a set, input/output control operation is started using specific IOCTL and Set GID, and DS with RUID and the reference which points out other required data (for example, a control flag, DS, etc.). For example, setting up a volume property by the sound card driver is inevitably accompanied by using the pinpointing RUID in the set which shows suitable GUID of the control flag which shows the property set IOCTL and set property operation for input/output control operation, and the property set with the volume set point, and a volume property, and the new volume set point.

[0081] for making a reference according to a type about the set supported -- null -- IOCTL (for example, Property IOCTL, Method IOCTL, or event enabling (IOCTL)) with the control flag which shows listing of GUID and the set supported specific set type is taken out as a part of I/O command, and the list of sets GUID supported is returned. In order to make a reference about the property, method, or event in a certain set supported, the control flag which shows Set GUID and listing of the element which set-type-IOCTL(s), and null-RUID(s) and is supported is used

with an I/O statement. The list of RUID supported is returned as a result of an I/O statement. From this list, a third party agent can judge which option element (in a certain case) of the set mounted is supported.

[0082] the specification of the set uniquely identified by GUID -- both a driver developer and a third party control agent -- although -- the mechanism which can be used as a mounting guide is document-ized. A third party developer will get to know the function of a certain driver based on the response to enquiry, and will know the knowledge programmed in advance based on an abstract set definition. Similarly, a driver developer can use an abstract set definition as a guide when mounting the set or set group who offers the function got to know to what kind of third party agent.

[0083] In order to offer the connect function currently explained here, the comp rye ANTO driver must be supporting a set of a certain kind. The following tables are supported in a property set format, and show some important information which can be used when this invention is mounted. The 2nd table shows the property about the actual contact pin instance created as a template using a specific contact pin factory about the property about the contact pin factory where the first table is mounted with a filter.

[0084]

[Table 1]

（表１）

フィルタ・プロパティとその使い方	
プロパティ	説明
接続ピン・ファクトリ	特定のフィルタで作成できる異種タイプの接続ピン・インスタンスをリストしている。各区別可能なタイプはピン・ファクトリと呼ばれる。なお、これはこのデバイスでインスタンス生成できる接続ピン・インスタンスの総数ではなく、オーディオ入力やオーディオ出力のように、ユニークな接続ピン・タイプの数である。
接続インスタンス	ある接続ピン・ファクトリの、すでに作成されたインスタンスの数と、その特定接続ピン・ファクトリ用にサポートされるインスタンスの最大数をリストしている。フィルタが実際に接続されるまで総数が決定できないときは、このプロパティはａ－１を返す。
データ・フロー	接続ピン・ファクトリがフィルタに対して取り得るデータ・フローの方向をリストしている（例えば、フィルタに入る方向、フィルタから出る方向、またはフィルタから出入りする方向）。

[0085]

[Table 2]

（表１のつづき）

通信	ある接続ピン・ファクトリの通信要求をＩＲＰの処理の観点からリストしている。一部の接続ピン・ファクトリは相互接続できないが、グラフ上のソース・ポイントを表すデータのファイル・ソースへの「ブリッジ(bridge)」のように、関連づけられた他の形態の制御メカニズムを有している。ブリッジ制御メカニズムは情報がストアされているファイル名を間接的に設定することを可能にする。 実施の形態では、エージェント（接続ピン・インスタンスを作るときどのピン・ファクトリを使用するかを判断する）は接続ピン・ファクトリの「なし」、「シンク」または入力、「ソース」または出力、「両方」および「ブリッジ」通信タイプの本来の意味を理解できなければならない。例えば、ソース接続ピン・インスタンスはシンク接続ピン・インスタンスなどへのハンドルまたは参照を必要とする。 通信タイプのコンテキストでは、シンクとソースとはＩＲＰを処理するときの接続ピン・インスタンスの処置に関する。シンクは処理するＩＲＰを受信するのに対し、ソースはＩＲＰを次の適切な処理コンポーネント上に引き渡す。 シンクでもソースでもなく、接続グラフのエンド・ポイントを表している通信タイプが２つある。 エンド・ポイントはデータが接続されたフィルタに出入りする場所を表している。「なし」とは接続タイプがインスタンシェイトできないことを意味するのに対し、ブリッジ通信タイプとは特定のプロパティを操作できるようにインスタンシェイトできるエンドポイントを意味する。例えば、ファイル・リーダの一部であるブリッジ・エンドポイントは処理されるデータをストアしているファイルのパスとファイル名を含んでいるプロパティをもっている可能性がある。

[0086]

[Table 3]

（表１のつづき）

データ範囲	接続ピン・ファクトリがサポートできる、可能な限りのデータ範囲をリストしている。関連すれば、データのフォーマットも含まれる。一実施の形態では、データ範囲の配列があとに続くカウントは接続ピン・タイプがサポートできるが、プロパティの一部として使用される。この実装においては、異なるデータ範囲が異なるメディアまたはインタフェースの下でサポートされる場合（下記参照）、異なる接続ピン・ファクトリが特定フィルタで利用でき、この相違を受け入れることができる。さらに、各データ範囲構造はビット数やチャネル数といった、フォーマット固有の詳細用に拡張可能である。接続ピン・インスタンスが使用する実際のデータ・フォーマットはインスタンスの作成時に設定される。データ範囲プロパティは、その実際のデータ・フォーマットが特定の接続ピン・インスタンス用にどのようにされるべきかを判断するとき使用され、サード・パーティ制御エージェントによってアクセスまたは照会される。
インタフェース	特定の接続ピン・ファクトリ上のサポートされるインタフェースを示す他の集合ＧＵＩＤをリストしている。インタフェースは接続ピン・ファクトリを通して通信できるタイプまたはデータのタイプである。例えば、ＭＩＤＩデータ、ＣＤミュージック、ＭＰＥＧビデオなどは、フィルタが処理できる特定の規則とフォーマットをデータがもっているという意味でインタフェースとなる。このようなインタフェースはデータを発信するためのプロトコルも含んでいる。インタフェースはそれが通信されるときの媒体から独立している。
媒体	特定の接続ピン・ファクトリ上のサポートされる媒体をリストしている。媒体とは、ＩＲＰベース、ソケットなどのように、情報が転送されるときの方法またはメカニズムである。インタフェースは種々の異なる媒体の上に定義できる。本明細書で説明している好適実施の形態と実装例では、ＩＲＰベースの媒体とファイル入出力ベースの媒体が使用されている。

[0087]

[Table 4]

（表１のつづき）

データ交差	データ範囲のリストが与えられているとき接続ピン・ファクトリによって作られた最初の受付可能または「最良」データ・フォーマットを返す。このアプローチは、サード・パーティ・エージェントが異なるフィルタを１つにチェインニングするときデータ要件を判断できるようにするために使用される。一実装例では、データ交差(data intersection) プロパティはデータ範囲のリストの制約が与えられているとき接続ピン・ファクトリによって作られた最良データ・フォーマットを判断するために使用される。データ範囲のリストは前述したように接続される別のピン・ファクトリでデータ範囲プロパティを使用して取得できる。 サード・パーティ制御エージェントは、データ・タイプの詳細を知らないので、ある接続ピン・ファクトリのデータ範囲リストを使用し、現行接続ピン・ファクトリで「最良」（例えば、最初の受付可能データ・フォーマット）を返すことができる。２つの交差接続ピン・ファクトリの範囲のセットを返すことができるが、最良フォーマットだけがドライバによって返される。このようにすると、サード・パーティ制御エージェントはこの「最良」データ・フォーマットをグラフ内の次のドライバに適用して、仮想的な(virtual) 接続集合を作成してから、実際に接続ピン・インスタンスの作成を開始し、フィルタのグラフ全体を１つに接続することができる。これにより、制御エージェントはユーザによって選択された特定フィルタ・グラフの実行可能性を評価し、ユーザに起こる可能性のある問題を指摘してから、実際にグラフを接続することができる。返されるデータ・フォーマットはフィルタですでに行われている接続が与えられているとき、使用可能なフォーマットで制限することもできる。 このプロパティは実際の接続が行われるまで特定のデータ・フォーマットが決定できないときや、交差が異なる接続ポイント上の複数のデータ・フォーマットに依存するときエラーを返すことができる。基本的には、交差情報が提供され、プロパティ自身はデータ・フォーマットを返す。

[0088]

[Table 5]

（表２）

接続ピン・インスタンス・プロパティとその使い方	
プロパティ	説明
状態	接続ピン・インスタンスの現状態を記述している。起こり得る状態には、停止、データ取得、データ処理、休止またはアイドルなどがある。状態は接続ピン・インスタンスの現モードを表し、ドライバの現在の機能とリソース使用状況を判断する。停止状態は接続ピン・インスタンスの初期状態であり、最小リソース使用状況のモードを表している。停止状態は、実行状態に到達するためにデータ処理のレイテンシ(latency) が最大であるポイントも表している。取得状態は、データがこの状態で転送されない場合でもリソースが取得されるモード（バッファ割当てなど）を表している。休止状態は最大リソース使用状況のモードと、実行状態に到達するまでの処理レイテンシがこれに応じて低いことを表している。データはこの状態で、転送または「プリロール(prerolled) 」できるが、これは実際には実行状態ではない。実行状態はデータが接続ピン・インスタンスで実際に消費または作成される（例えば、転送され、処理される）モードを表している。フィルタと基礎となるハードウェアの目的に応じてカスタム・プロパティを使用すると、コントロールの解像度を向上することができる。例えば、外部レーザ・ディスク・プレイヤをスピンアップさせるには、そのクラスに固有のある種のカスタム「モード」プロパティを設定することになる。このプロパティをセットすると、モードの効果に応じてデバイスの状態も変更されるが、必ずしもそうとは限らない。

[0089]

[Table 6]

（表２のつづき）

優先度	フィルタによって管理されるリソースへのアクセスを受け取るための接続ピン・インスタンスの優先度を記述しており、リソース割当ての調停(arbitration) においてフィルタによって使用される。このプロパティがサポートされていれば、サード・パーティ制御エージェントは限定リソースをこの特定接続およびインスタンスと共用できる他のすべてのドライバの他のすべての接続ピン・インスタンスに相対的な特定ピン・インスタンスの優先位置を指定することができる。この優先度プロパティが実装されていると、エージェントは単一優先度クラス内の優先度をもっときめ細かくチューニングすることができる。例えば、優先度はある種のサブクラスを関連づけることができる。同一リソースを競合する２つのドライバが同一優先度クラスをもっていれば、サブクラス優先度は２ドライバ間のリソース割当てを決定するために使用される。サブクラス優先度も同一であれば、調停により、最初の接続ピン・インスタンスがリソースを受け取ることになる。
データ・フォーマット	接続ピン・インスタンスのデータ・フォーマットを取得または設定するために使用される。

[0090] In order to create connection between different drivers so that I may be understood, if this field that is not what the above-mentioned table is mere instantiation and is limited to this becomes a familiarity person, it is possible to mount the property with which a large number differ, and a schema. the capacity to mount the property set with same driver manufacturer or development group who one important element is a standardization multiplier (standardization factor), and is different -- **** -- since it is, it is enabling it to create the driver which can interconnect.

[0091] Another useful property set gives the topology information about the internal relation of the contact pin factory of the input on a certain filter, and an output. Not only the relation between the input pin factory on a certain filter, and a corresponding output pin factory but this information has indicated processing [what kind of type] is performed between the pin factories of an input and an output. As an example of the processing performed, there are different data conversion, data elongation, an echo denial (cancellation), etc. If you use such information by the automation filter graphing function which creates an actual contact pin instance and connection after it traces the connection pass on an assumption using two or more filters, it is convenient. Fundamentally, topology information explains the internal structure of a filter and opens this to enquiry from a third party agent through a property set mechanism.

[0092] Therefore, the comp rye ANTO driver mounts the specified property set. Therefore, a third party control agent can perform enquiry and a setup to a comp rye ANTO filter, when it turns out that a certain property set is supported. A whole target is acquiring sufficient information about the approach of connecting a different filter to one and making a filter graph.

[0093] Although the minimum function is mounted and the system of a comp rye ANTO driver can be supported if a general-purpose set mechanism is used, expandability is unrestricted even in such a case. as long as, as for the set, the specific set is mounted -- mutual -- in order to

create the system of the driver which can interconnect [that it is operational and], a definition can be given in the specification which can be independently coded by the driver developer from whom a large number differ. Furthermore, the property of the option which it not only can define the indispensable property which must be supported, a method, and an event, but can be mounted according to driver ability and extension, a method, and an event can also be defined as this specification. It is also possible to incorporate an additional function by defining the set with an original driver developer other than the fundamental similarity required of the minimum, and assigning these GUID.

[0094] Next, it explains with reference to drawing 9 and drawing 10 . Drawing graphic-form-izes the process which connects two Carnell mode filters, and shows it. Drawing 9 is a logic-block explanatory view, and each filter instance and a contact pin instance are expressed with the file object there. Drawing 10 is a flow chart which shows the step when creating a file object and suitable connection.

[0095] It starts from step 144 and the instance of a filter A146 and the instance of a filter B148 are created by the user Mohd agent. These are created with a specific device using the standard file system API which creates a file. It is because it refers to the function of each filter about the contact pin factory which that the filter A146 and the filter B148 serve as a comp rye ANTO filter or a driver mounted the property with these suitable, the method, and the event set, supported creation of a contact pin instance, and was defined the set supported and for [its] filters.

[0096] A third party control agent refers to a filter A146 and a filter B148 at step 150, respectively, and judges the attribute of the contact pin instance which can be created from an available contact pin factory and available it. These attributes have the connection format and data format according to type of each pin factory of each filters 146 and 148, as mentioned above. Enquiry is held below using the set base enquiry mechanism explained in detail.

[0097] After finishing enquiry of the above-mentioned information, a third party control agent opts for the optimal connection format based on the range of the data format for which it referred before, and a connection format. This decision is made at step 152 and it can be made to do usage depending on which third party agents differed the same filter according to the requirement of the selected connection pass. A third party control agent uses a data crossover property (data intersection property), topology information, and a contact pin factory with both filters, and determines the best selection approach of a data format and connecting arrangement according to the actual filter graph created.

[0098] The input filter pin instance 154 uses the optimal detection information judged at step 152, and is created by the third party agent at step 156. Since the input pin instance 154 is a file object, a handle is returned from a creation process, but this can be used in order to send input/output request to the input instance 154. Furthermore, although creation of the input pin instance 154 is inspected in effectiveness, routing and the validation mechanism which were mentioned above with reference to drawing 4 - drawing 6 , drawing 7 , and drawing 8 are used for this creation.

[0099] In order to complete connection, the output pin instance 158 is created at step 160 as a parameter in NtCreateFile Kohl using the handle of the input pin instance 154 created before. As effectiveness that the output pin instance 158 does in this way, and is created, internal IRP stack structure is created using a system file function manager and an input/output management function. Since it becomes possible to carry out consecutive processing of the original write-in command to the contact pin instance connected by various approaches in suitable sequence with a filter, the immediate-data flow between different filters is easy-ized by this stack structure. In order to perform this, it is required to create the input pin instance before the related output pin instance which supplies an input pin instance.

[0100] The stack depth parameter of a device object controls how many a stack location is created to IRP sent to this driver. Although it is assumed that the number of stack depth parameters is one when a device object is made for the first time, it is also possible to change later according to whether chaining of two or more drivers is carried out to one. In the present system, if required, this change will be made, when an output pin instance changes in "acquisition" or other condition from an initial "halt" condition. The state transition of a contact pin instance is a mechanism which determines the right stack depth parameter information for creating IRP correctly and processing it.

[0101] In order to assign correctly the contact pin instance set by which chaining was carried out, it is necessary to make it change so that it may come out of a contact pin instance from a idle state in specific sequence. that is, -- from the last input pin instance (this example input pin instance 154) -- starting -- output (for example, it connected) pin instance (this example output pin instance 158) of relation up to -- it is necessary to return to hard flow continuously If chaining of many filters is carried out to one, the input pin instance of the deepest filter or a bridge is the initiation point of transition, and it must build continuously to hard flow until the initial output pin instance on a bridge or a filter is set up. the transition which in other words comes out of a idle state -- a chain -- ****** -- it is carried out like and must be made to have to obtain the stack size for which each contact pin instance is needed after a front contact pin instance As an example of representation, although it is not necessary to necessarily do so, and a contact pin instance changes from a idle state to an acquisition condition, the same purpose as the transition of the following explanation to which the transition to an acquisition condition comes out of a idle state about adjustment of a stack depth parameter for convenience is achieved.

[0102] After all pin instances are in an acquisition condition, stream read and writing can be taken out to a filter graph. In addition, the system which explains what should be observed here

can make connection of a related input and an output pin instance in every sequence, and is having to perform only transition from a idle state first from a bottom up system, i.e., the deepest thing. Furthermore, reconstruction of a filter graph is attained so that it can change after initial creation. When a change is made, a state transition is performed only by the contact pin instance in a idle state, and right stack depth parameter information needs to be made to be acquired.

[0103] The contact pin factory found out on a filter expresses the location where a filter can consume and/or create data in a specific format. For example, a specific contact pin factory can support the data format from which some, such as a 16 bits 44kHz PCM audio and a 8-bit 22kHz PCM audio, differ. As mentioned above, about different functions, such as a contact pin factory and its data format, a reference can be made from a filter using a suitable property set mechanism and a system input/output function. An actual contact pin instance is created based on the information received from the pin factory.

[0104] If single stream writing or stream read operation is taken out by the user Mohd agent, in the streaming environment where consecutive processing of data is performed through the connected filter, two Maine methods for IRP control can use it as a part of NETIBU function of NT operating system. individual by the 1st method -- IRP is created with each filter, is sent to the following filter and processed, and this filter creates new IRP, goes down a chain, and is processed one after another. By other methods, single IRP is used and it is delivered between continuous filters using the standard procedure prepared in order to interact with an I/O manager. For the interconnect sequence between filters not being important for when the 1st method which creates new IRP for every continuous filter on a chain is used, a filter is the destination (destination) of IRP. It is because it can act as Kohl of the I/O manager and he can be seen off in the filter of assignment of IRP that what is necessary is just to know. A thing important when the reuse of the IRP is carried out is that even the filter which created IRP for processing processes

even the first filter which is performed so that it may begin from the last filter with which transition of the contact pin instance from a idle state receives Reuse IRP, and receives Reuse IRP to hard flow.

[0105] The gestalt and the example of mounting of this operation of an interconnect Carnell mode filter mitigate the complexity of driver development using IRP common use, make it possible to create a stronger driver, and have the advantage of increasing the efficiency of processing. The pin instance state-transition pass of a "bottom-up" guarantees that right stack sequence is created by IRP processed by the continuation driver, and has the stack depth parameter set with each suitable driver object. Furthermore, the present condition voice of a receiving-side input pin factory is checked in order to confirm whether the state-transition sequence was followed correctly. From such a reason, the communication link property of a specific contact pin factory judges the direction of a flow which may happen, and when distributing the state transition of a contact pin instance correctly, it supports it.

[0106] When creating an output pin instance (or IRP source), the reference to the file object showing the input pin instance on another filter (or IRP sink (sink)) is handed over as some NtCreateFile Kohl. A suitable creation handler is performed as the multiplexing dispatch function, and the device object / file object hierarchy were used and mentioned above. This creation handler can access the device object of a filter (for example, filter B148 of drawing 9) with an input pin instance through an input contact pin instance file object (for example, input pin instance 154). A front stack depth parameter can be read and the stack depth parameter of the device object of a filter with an output pin instance is increased from a device object. For example, the device object relevant to a filter A146 has the stack depth parameter which is a device object relevant to a filter B148, and is increased in the connection shown in drawing 9 . Usually this is performed at the time of the transition which comes out of a idle state, and while a

contact pin instance is in a idle state, routing of the IRP is not carried out.

[0107] When a filter processes IRP, it knows which stack frame or location should be accessed by using it with reference to the stack depth parameter of a related device object. [in an IRP stack including the information specified for / the / specific filters] Furthermore, the present filter prepares IRP for the next filters on a processing chain by reducing the stack depth parameter of a device object and positioning in the IRP stack location of the following filter.

[0108] It takes charge of a filter code preparing the next location in an IRP stack, and acting as Kohl of the I/O manager, and passing IRP to the following filter as specified. If it does in this way, it can specify whether which file object showing a specific contact pin instance receives IRP and associated data, and a filter processes. Therefore, standard-input/output manager Kohl like IoAttachDevice for carrying out the stack of each device object for sequential processing of IRP is not used.

[0109] What should be observed is not meaning creating connection between contact pin instances creating a device object new since the connection is expressed. The single device object used as a foundation is used in order to support the instance of a filter, and all the contact pin instances on the filter. Although concrete information required for right data processing is saved in the context area of a file object and context information is left behind as it is, non-page memory usage is maintained at the minimum. Although what should furthermore be observed has explained the medium of the IRP base, it is being able to use other media for the communication link between interconnect filters. Direct function Kohl in non-host hardware and the communication link between hardware is one of such things.

[0110] Next, drawing 11 , drawing 12 , and drawing 13 are referred to, and it is drawing 1 .

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is the data flow Fig. of the conventional technique which shows the interconnect filter which carries in sound data from a disk file in response to directions of a control agent, processes sound data with a certain gestalt, carries out the rendering of the sound data, and is reproduced from a loudspeaker, and the system of a driver.

[Drawing 2] The system and the purpose which are shown in drawing 1 are the same, read sound data in a disk driver, process that data, carry out the rendering of that data, it is made audible from a loudspeaker, and a processing filter and a rendering are drawings showing the system by this invention carried out as [process / in response to directions of a control agent / in this case / by the interconnect Carnell Mohd driver].

[Drawing 3] It is drawing showing the perpendicular relational model which is created with an operating system and shows the relation between the driver object used, a device object, and a file object.

[Drawing 4] It is the logic-block Fig. of a driver object, and is drawing showing the logical relation between the DS for transmitting a message to a suitable process handling code according to the system of this invention, and carrying out validation of the creation of a new file object, and a program code.

[Drawing 5] It is the logic-block Fig. of a device object, and is drawing showing the logical relation between the DS for transmitting a message to a suitable process handling code according to the system of this invention, and carrying out validation of the creation of a new file object, and a program code.

[Drawing 6] It is the logic-block Fig. of a file object, and is drawing showing the logical relation between the DS for transmitting a message to a suitable process handling code according to the

system of this invention, and carrying out validation of the creation of a new file object, and a program code.

[Drawing 7] It is the flow chart which shows routing, the initial setup of validation KOMPONENTORI, and processing of the I/O message by the Carnell Mohd driver.

[Drawing 8] It is the detail flowchart which shows processing of a control agent, routing, a validation mechanism, and the concrete creation handler routine that creates a new file object.

[Drawing 9] It is the logic diagram showing the level relation between the connection filters performed by the approach which had connection standardized with an operating system using file object structure.

[Drawing 10] The Carnell mode filter or driver of drawing 9 is created, and in order to connect, it is the flow chart which shows the processing step taken by user Mohd's control agent, it connects in order to process the input/output request received from the control agent, and signs that it is continued between the drivers (filter) from which the processing differs are shown.

[Drawing 11] It is the outline logic diagram showing the Carnell Mohd driver for mounting the system is used in order to create the chain of the Carnell mode filter in response to directions of user Mohd's control agent, reads sound data in a hard drive, processes the data with the Carnell mode filter, carries out the rendering of the data, and it is made to be heard from a loudspeaker, and connection.

[Drawing 12] It is the outline logic diagram showing the Carnell Mohd driver for mounting the system is used in order to create the chain of the Carnell mode filter in response to directions of user Mohd's control agent, reads sound data in a hard drive, processes the data with the Carnell mode filter, carries out the rendering of the data, and it is made to be heard from a loudspeaker, and connection.

[Drawing 13] It is the flow chart which shows the processing step for creating an interconnect

Carnell Mohd driver to the systems shown in drawing 11 and drawing 12 .

[Description of Notations]

44 Control Agent

46 Disk Drive

48 Disk Driver

50 Reader Driver

52 Decompressure Driver

54 Effectiveness Filter

56 Effectiveness Processor

58 Rendering Driver

62 Loudspeaker

64, 76, 80 Driver object

66 Device Object

70, 72, 74, 90 File object

78 Multiplexing Dispatch Function in General

82 Device Extension Area

84,100 File type validation table

86,102 File object type

88 Creation Handler

92 File Context Area

94 IRP Demand Handler Table

96 IRP Demand

98 Handler

104 Reference

146 Filter A

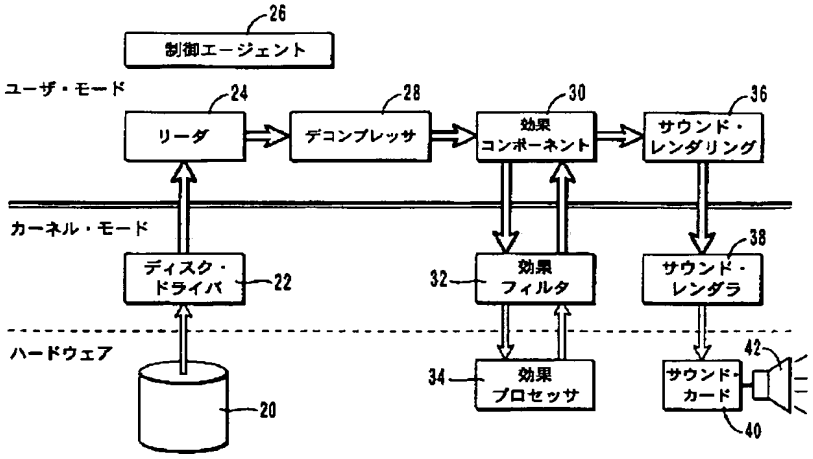
148 Filter B

154 Input Filter Pin Instance

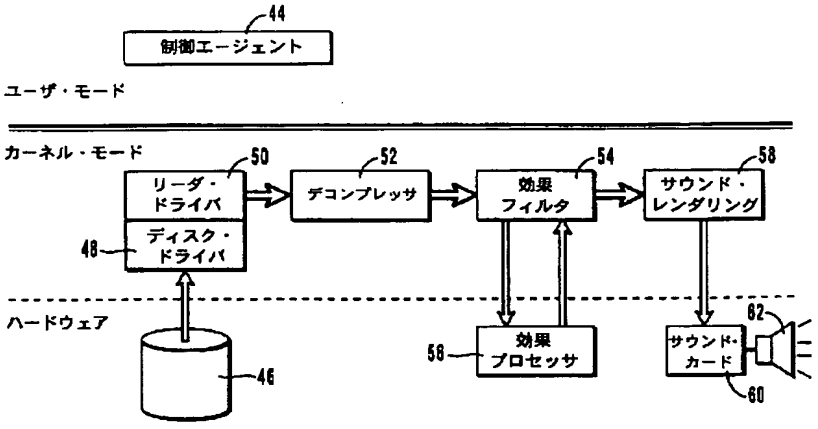
158 Output Pin Instance

DRAWINGS

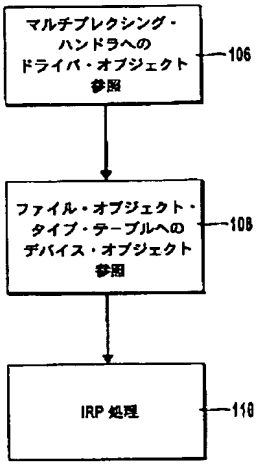
[Drawing 1]



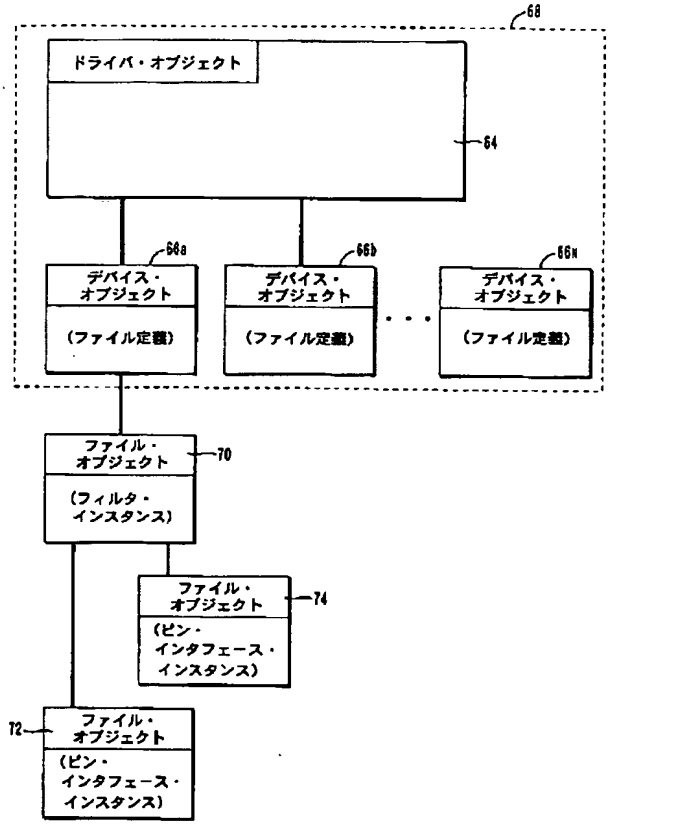
[Drawing 2]



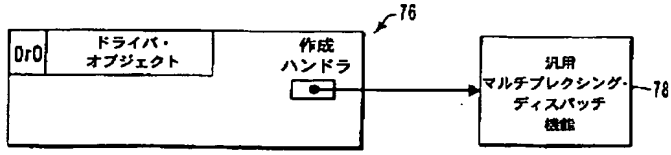
[Drawing 7]



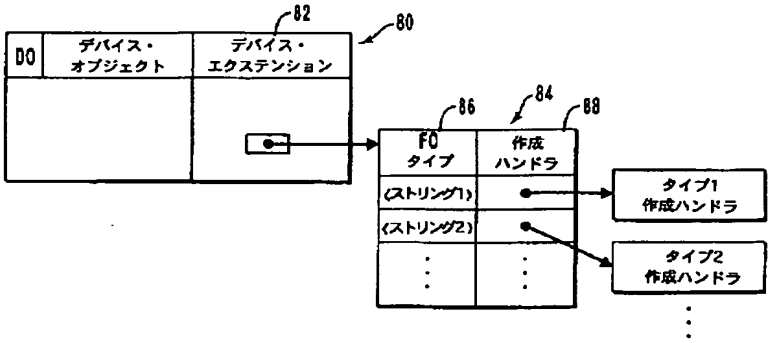
[Drawing 3]



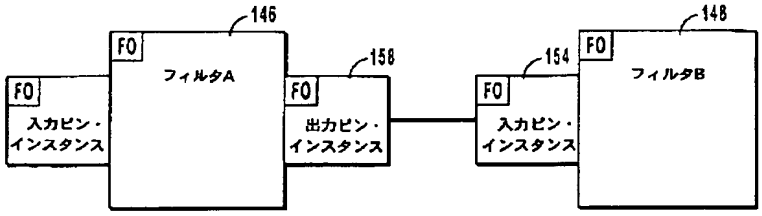
[Drawing 4]



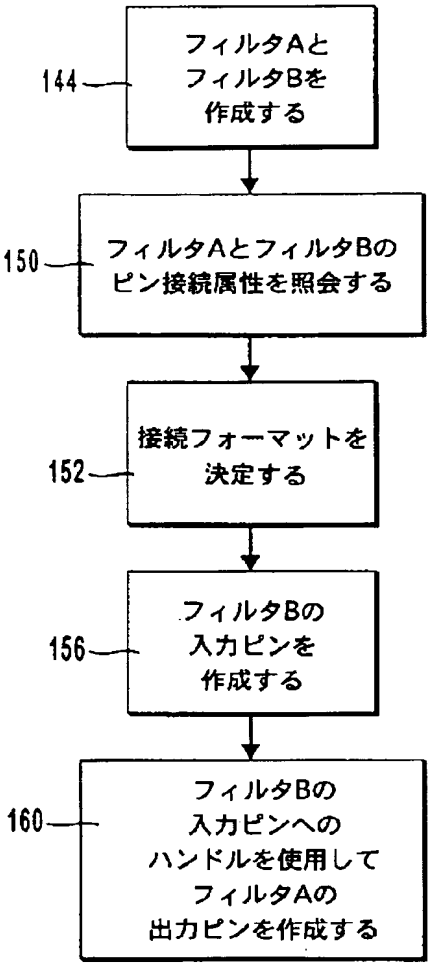
[Drawing 5]



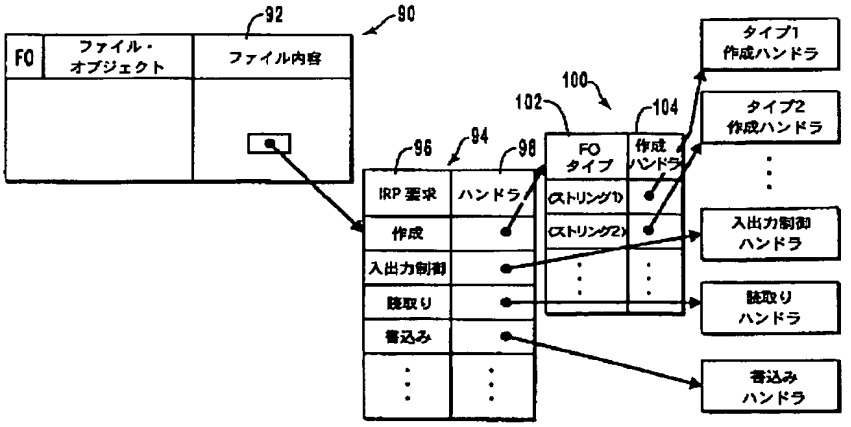
[Drawing 9]



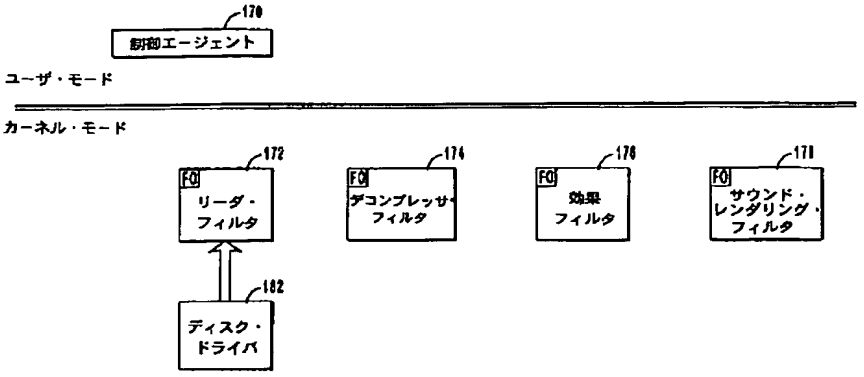
[Drawing 10]



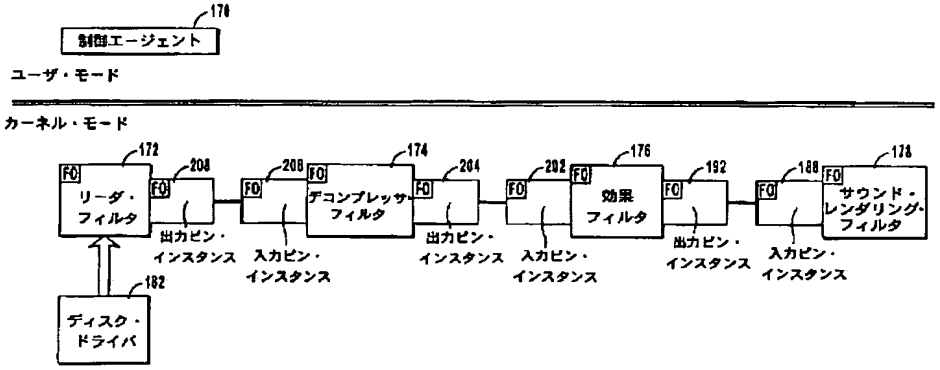
[Drawing 6]



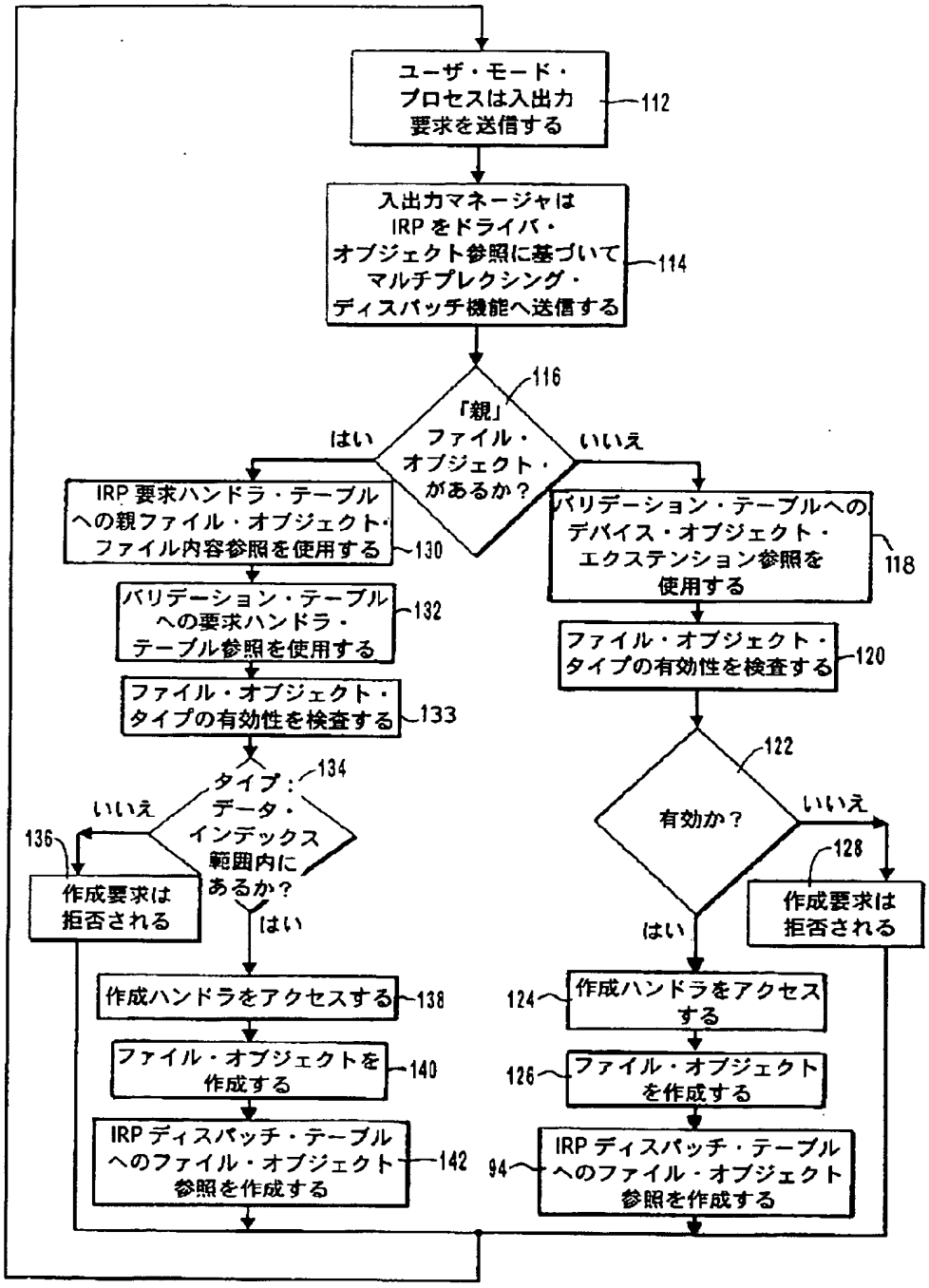
[Drawing 11]



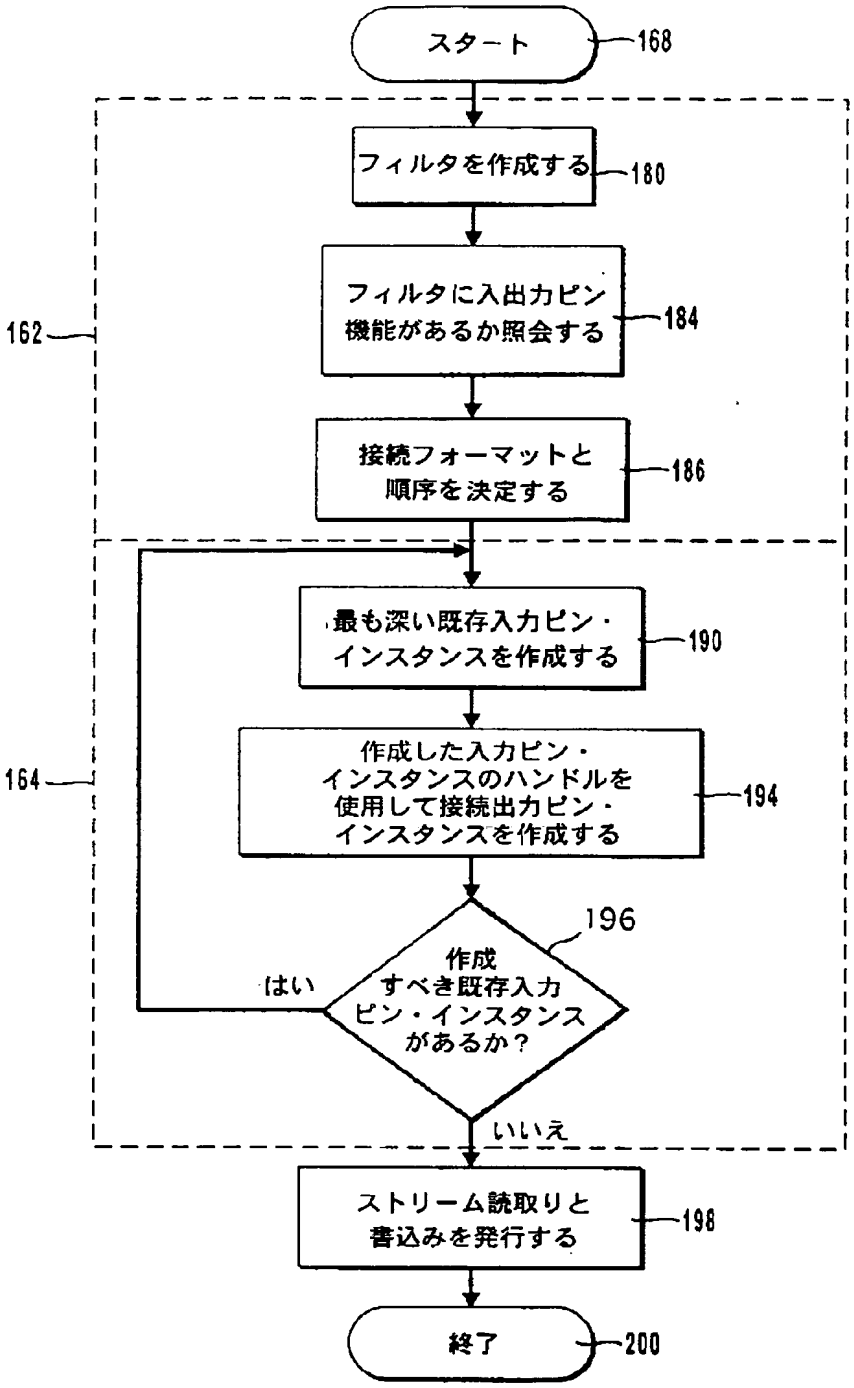
[Drawing 12]



[Drawing 8]


ユーザ・モード・プロセスは入出力要求を送信する
112
入出力マネージャはIRPをドライバ・オブジェクト参照に基づいてマルチプレクシング・ディスパッチ機能へ送信する
114
116
「親」ファイル・オブジェクト・があるか？
はい
いいえ
IRP要求ハンドラ・テーブルへの親ファイル・オブジェクト・ファイル内容参照を使用する
130
バリデーション・テーブルへのデバイス・オブジェクト・エクステンション参照を使用する
118
バリデーション・テーブルへの要求ハンドラ・テーブル参照を使用する
132
ファイル・オブジェクト・タイプの有効性を検査する
120
ファイル・オブジェクト・タイプの有効性を検査する
133
122
有効か？
いいえ
はい
134
タイプ：データ・インデックス範囲内にあるか？
いいえ
はい
136
作成要求は拒否される
128
作成要求は拒否される
作成ハンドラをアクセスする
138
124
作成ハンドラをアクセスする
ファイル・オブジェクトを作成する
140
126
ファイル・オブジェクトを作成する
IRPディスパッチ・テーブルへのファイル・オブジェクト参照を作成する
142
94
IRPディスパッチ・テーブルへのファイル・オブジェクト参照を作成する

[Drawing 13]


スタート
168
フィルタを作成する
180
フィルタに入出力ピン機能があるか照会する
184
162
接続フォーマットと順序を決定する
186
最も深い既存入力ピン・インスタンスを作成する
190
作成した入力ピン・インスタンスのハンドルを使用して接続出力ピン・インスタンスを作成する
194
164
196
作成すべき既存入力ピン・インスタンスがあるか？
はい
いいえ
ストリーム読取りと書込みを発行する
198
終了
200